SunFounder PiDog Kit

Release 1.0

sunfounder

Jun 05, 2026

CONTENTS

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

-

Thank you for choosing our .

Note: This document is available in the following languages.

-
-
-
-
-
-

-

Please click on the respective links to access the document in your preferred language.

PiDog is a Raspberry Pi pet robot with aluminum alloy structure. It can act as a mechanical pet, show cuteness to you, and interact with you.

It is equipped with a camera module, which can perform color recognition, face detection and other projects; 12 metal gear servos support it to walk, stand, sit, shake its head, and pose in various poses; The ultrasonic module on the head enables it to quickly detect obstacles ahead; Special touch sensors allow it to respond to your touch; The Light Board on the chest can emit colorful light effects, and with the speaker equipped with the robot HAT, PiDog can express emotions such as happiness and excitement. In addition, PiDog is also equipped with a sound direction sensor and a 6-DOF IMU module to realize more complex and interesting usage scenarios.

If you have any questions, please send an email to service@sunfounder.com and we will respond as soon as possible.

The product is available in two versions: Standard and V2.

Standard Version

Designed for compatibility with most Raspberry Pi models, this version offers full functionality of the kit. Notably, its servo motors and Robot HAT are not compatible with Raspberry Pi 5, making the standard version unsuitable for use with this model.

V2 Version

This upgraded iteration features an updated Robot HAT and servo configuration, enabling compatibility with Raspberry Pi 3/4/5 and Zero 2W. The enhancements allow expanded creative applications on Raspberry Pi 5 while maintaining backward compatibility with earlier models.

Key technical distinctions:

- Hardware iteration: V2 redesigns servo drivers and HAT circuitry for RPi5's GPIO voltage requirements
- Model coverage: Standard supports RPi 3B+/4B/Zero 2W; V2 adds RPi5 support
- Power management: V2 optimizes power delivery for RPi5's higher current demands

Content

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

CHAPTER

ONE

ASSEMBLE VIDEOS

Before assembling the PiDog, please first verify that all parts and components have been included. If there are any missing or damaged components, please contact SunFounder immediately at service@sunfounder.com to resolve the issue as soon as possible.

Mount Raspberry Pi Zero 2W on PiDog

If your mainboard is a Raspberry Pi Zero 2W, here are the steps to install it on the PiDog.

Note: The assembly steps in the video may differ slightly from the printed instructions you have. Please prioritize following the printed instructions. If any steps are unclear, you can refer to the video for further clarification.

Afterward, you can continue following the instructions in the video below from **2:28** onwards to assemble it.

Assembly Tutorial Video(Raspberry Pi 4/3/1 Model)

This video will walk you through the process of assembling your robot from scratch.

Note: The assembly steps in the video may differ slightly from the printed instructions you have. Please prioritize following the printed instructions. If any steps are unclear, you can refer to the video for further clarification.

In this tutorial, you will learn:

- **Preparation**: We'll introduce you to all the tools and parts needed, ensuring you're fully equipped before starting the assembly.
- **Assembly Steps**: We'll demonstrate each assembly step in a systematic manner.
- **Tips and Considerations**: Throughout the process, we'll share essential tips and tricks to help you avoid common mistakes and ensure your robot operates smoothly.
- **Zeroing a Servo**: Before fixing each servo, it needs to be zeroed first. The steps for zeroing are to first install the Raspberry Pi OS, then install the required modules, and then run a script (set the angle of all PWM pins to 0). After that, plug in the servo wire to zero the servo.

The assembly process for Pidog is quite long, so we have divided it into two videos. The first video covers assembling Pidog's body and four legs.

The second video covers assembling the head and calibration.

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

CHAPTER

TWO

PLAY WITH PYTHON

If you want to program in python, then you will need to learn some basic Python programming skills and basic knowledge of Raspberry Pi, please configure the Raspberry Pi first according to *1. Quick Guide on Python*.

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

2.1 1. Quick Guide on Python

This section is to teach you how to install Raspberry Pi OS, configure wifi to Raspberry Pi, remote access to Raspberry Pi to run the corresponding code.

If you are familiar with Raspberry Pi and can open the command line successfully, then you can skip the first 3 parts and then complete the last part.

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

2.1.1 What Else Do You Need?

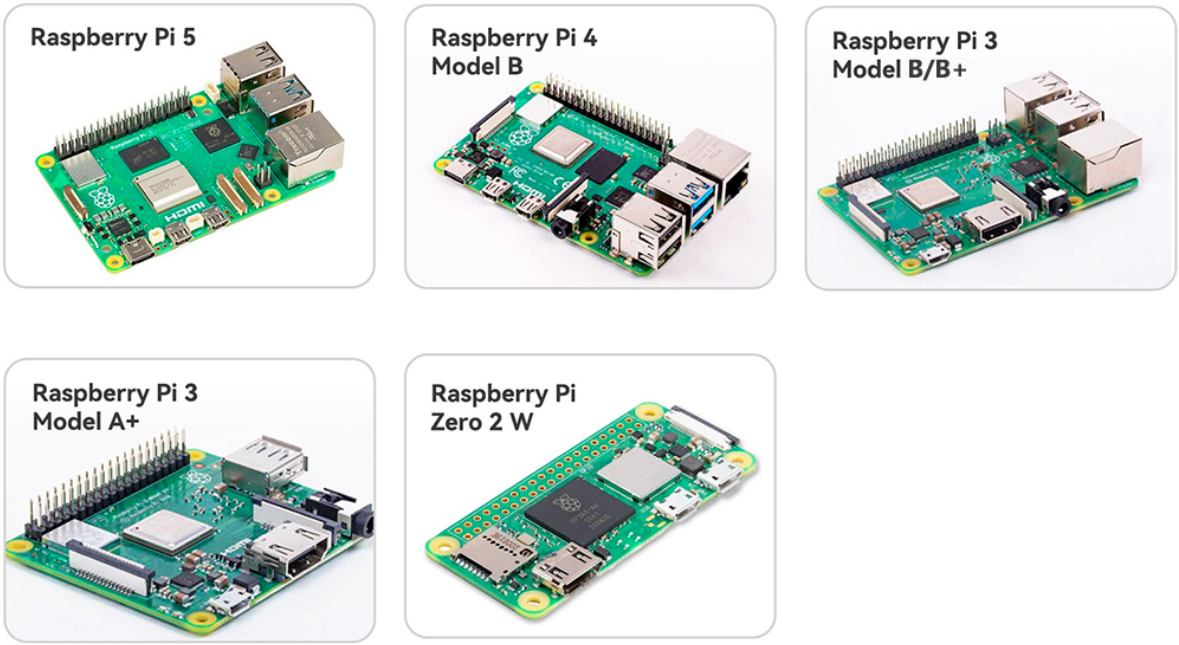
Before we start playing with this kit, let's prepare the essential hardware.

Required Components

- **Raspberry Pi**

  The Raspberry Pi acts as the **brain**, handling all computing, sensing, and control tasks.

  – **Compatible models**: Raspberry Pi 5, Raspberry Pi 4, 3, or Raspberry Pi Zero 2W



- **Power Adapter**

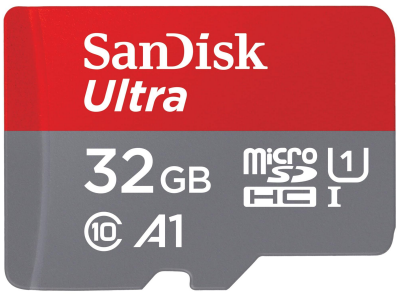
  Prepare a suitable power supply based on your Raspberry Pi model:

  - **Raspberry Pi 5**: 5V 5A USB-C (recommended: official 27W PD power supply).
  - **Raspberry Pi 4**: 5V 3A USB-C.
  - **Raspberry Pi 3B/3B+**: 5V 2.5A Micro-USB.
  - **Raspberry Pi Zero 2W**: 5V 2A Micro-USB.

  Using a stable power source helps prevent undervoltage and ensures reliable operation.

- **Micro SD Card**

  The Raspberry Pi does not have a built-in hard drive. It boots and stores all files on a **Micro SD card**.



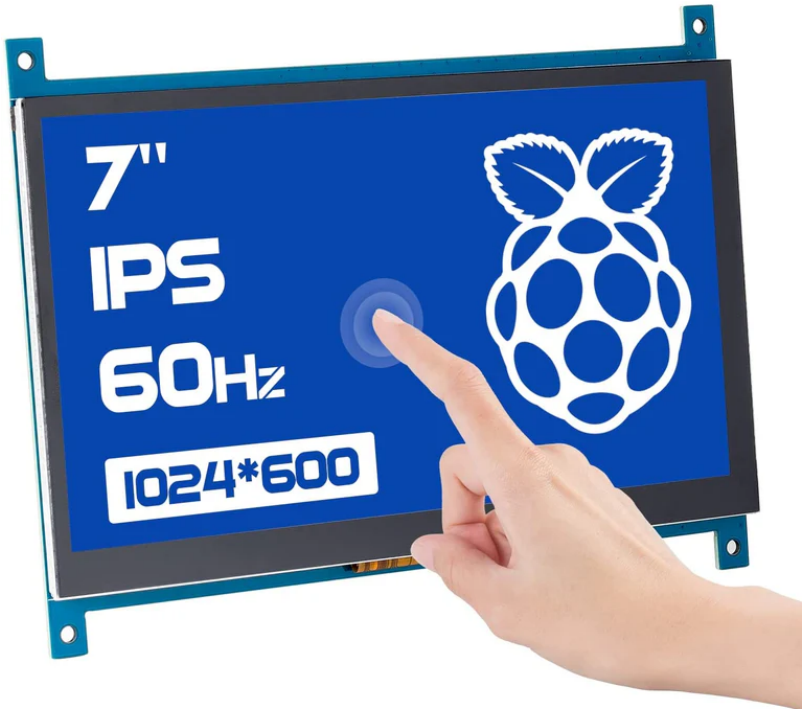
  - Minimum: **16GB**
  - Recommended: **32GB** for better stability
  - Brand: Use reliable options such as **SanDisk** or **Samsung** to avoid read/write errors

Optional Components

Although not strictly required, the following peripherals will greatly improve your learning and debugging experience:

- **Monitor (HDMI or TV)**

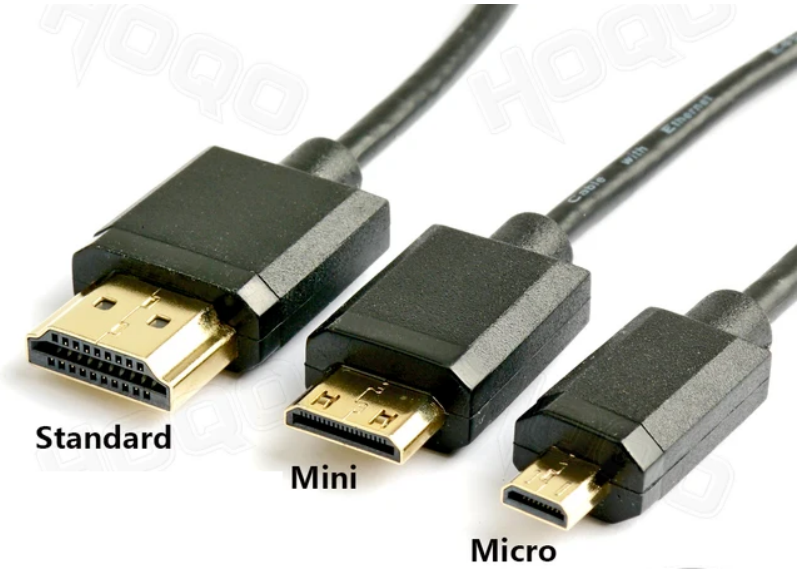
  For beginners, we strongly recommend a display with an HDMI input, so you can easily configure Raspberry Pi OS and run graphical programs.



- **HDMI Cable (Standard / Mini / Micro)**

  Different Raspberry Pi models use different HDMI connectors, be sure to check your Pi model and prepare the correct cable.

  - **Raspberry Pi 4 / 5**: Micro HDMI
  - **Raspberry Pi 3**: Standard HDMI
  - **Raspberry Pi Zero 2W**: Mini HDMI



- **Keyboard & Mouse**

  Very useful during the initial setup of Raspberry Pi OS. Later, you may switch to remote access (SSH/VNC), but for beginners we recommend preparing a basic USB or wireless set.

Tips for Preparation

- If you purchased this kit, most accessories are included, but you still need to prepare the Raspberry Pi board, Micro SD card, and power adapter separately.
- Not sure what to buy? The most stable and universal choice is: **Raspberry Pi 4/5 (2GB) + Official Power Supply + 32GB Micro SD card**.

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.

- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

2.1.2 Installing the Operating System

Before using your Raspberry Pi, you need to install **Raspberry Pi OS** onto a microSD card. This guide explains how to do that using **Raspberry Pi Imager** in a simple, beginner-friendly way.

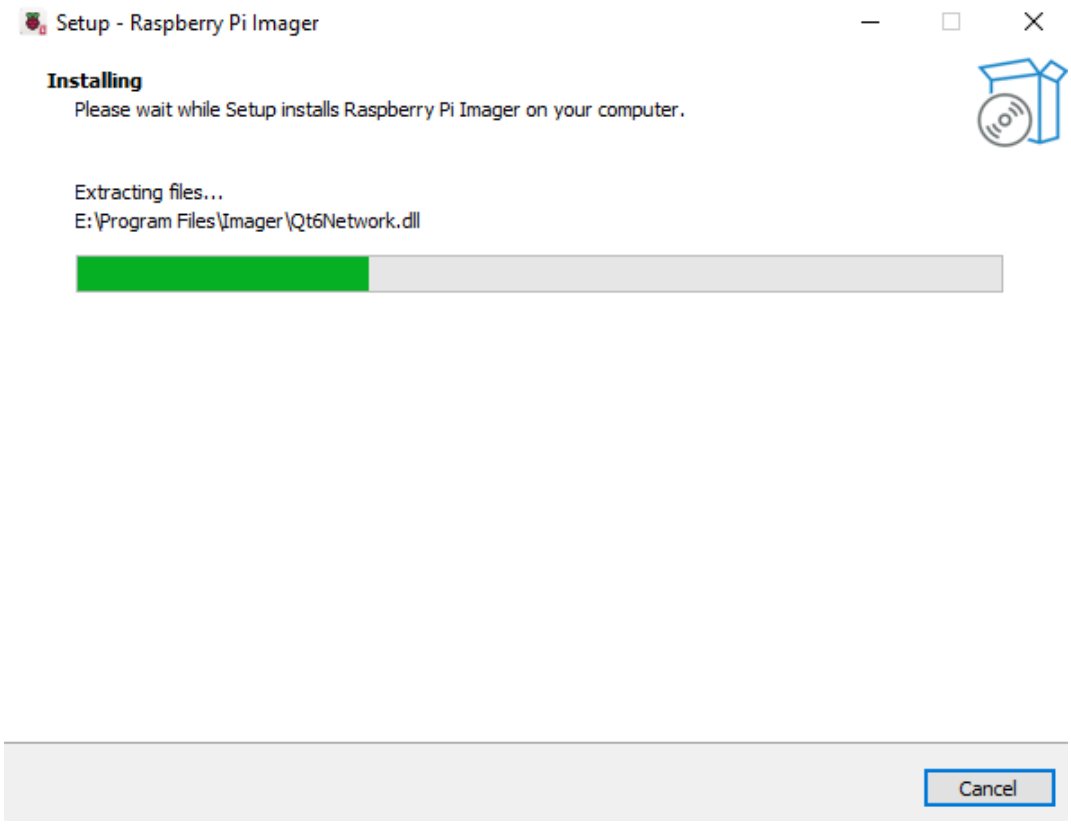
Required Components

- A computer (Windows, macOS, or Linux)
- A microSD card (16GB or larger; recommended brands: SanDisk, Samsung)
- A microSD card reader

Install Raspberry Pi Imager

1. Visit the official Raspberry Pi Imager download page: . Download the correct installer for your operating system.

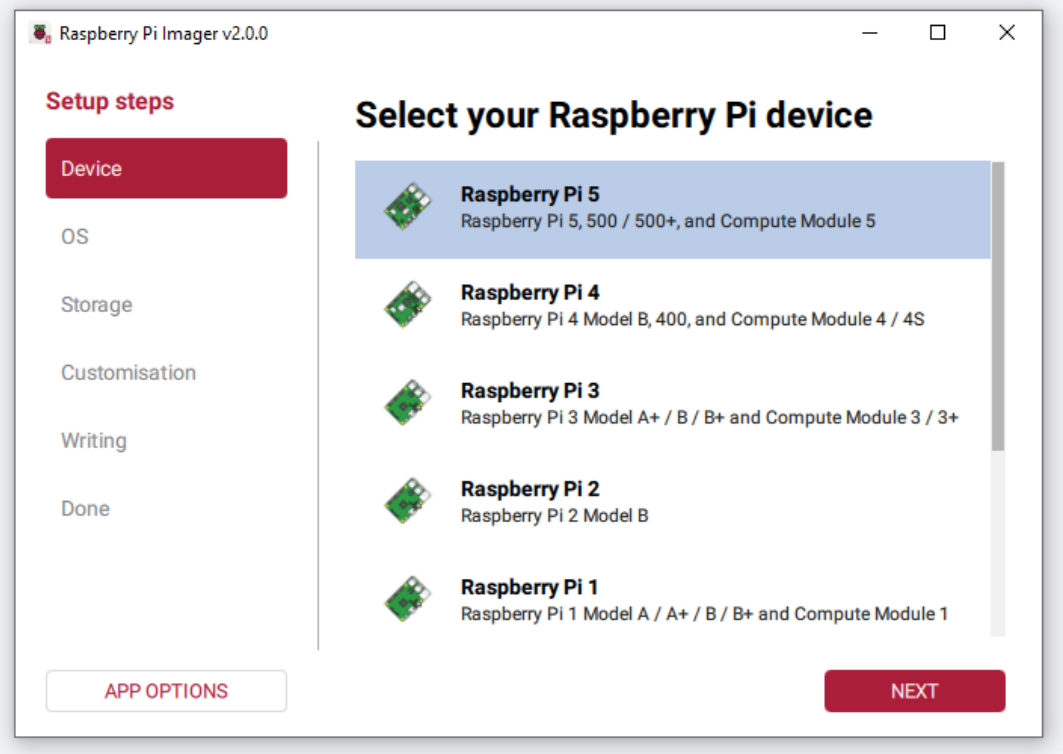
2. Follow the installation prompts (language, install path, confirmation). After installation, launch **Raspberry Pi Imager** from your desktop or applications menu.



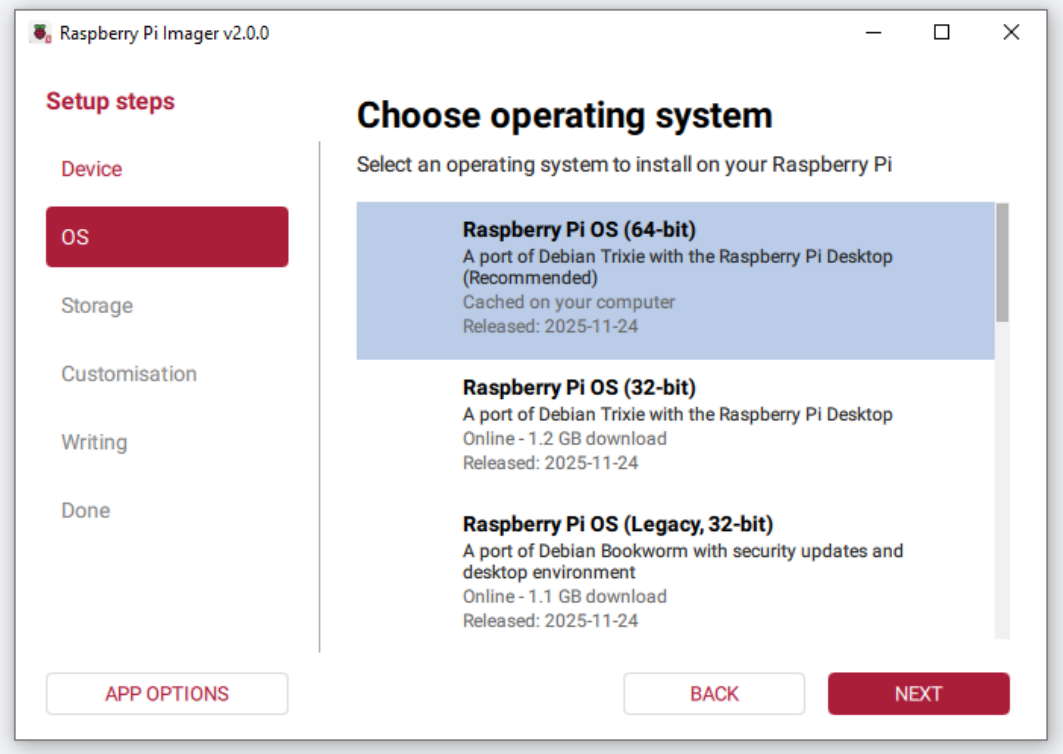
Install the OS to the microSD Card

1. Insert your microSD card into your computer using a card reader. Back up any important data before proceeding.

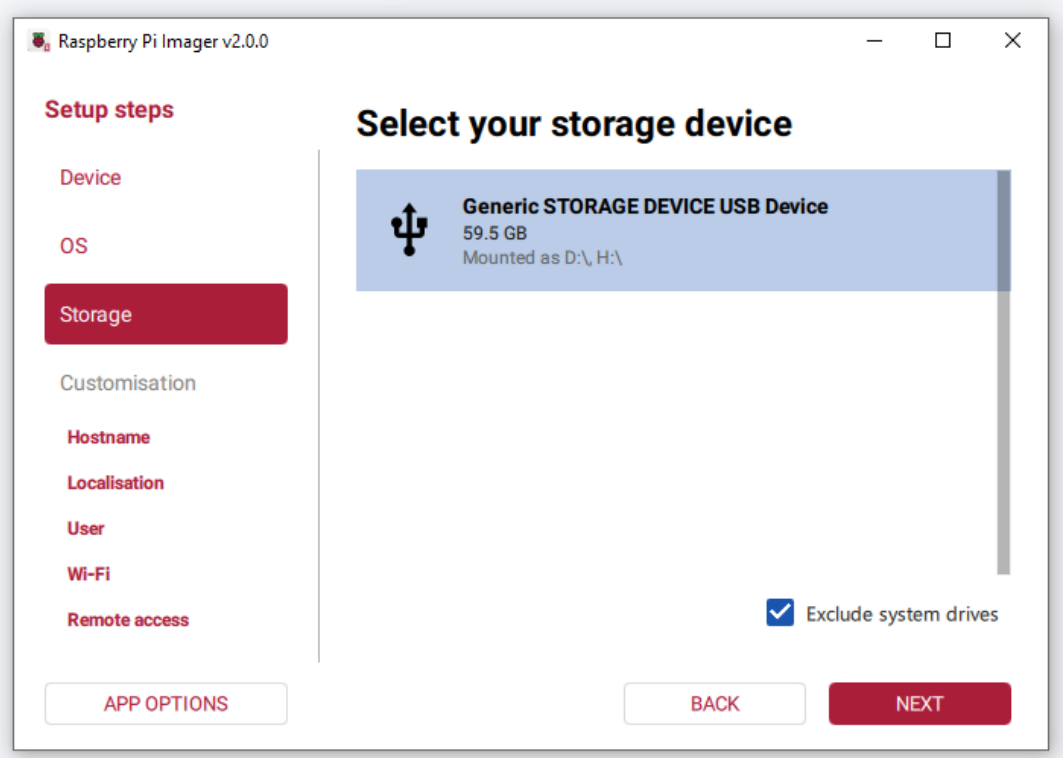
2. When Raspberry Pi Imager opens, you will see the **Device** page. Select your Raspberry Pi model from the list (e.g., Raspberry Pi 5, 4, 3, or Zero 2W).



3. Go to the **OS** section and choose the recommended **Raspberry Pi OS (64-bit)** option.

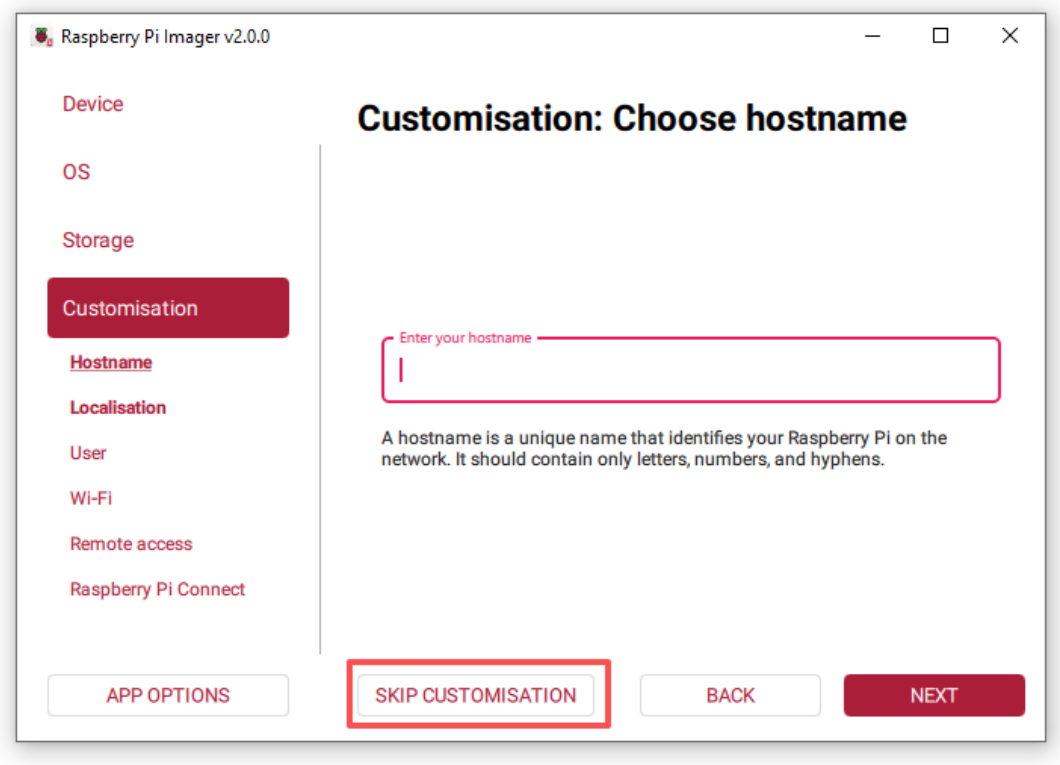


4. In the **Storage** section, select your microSD card.

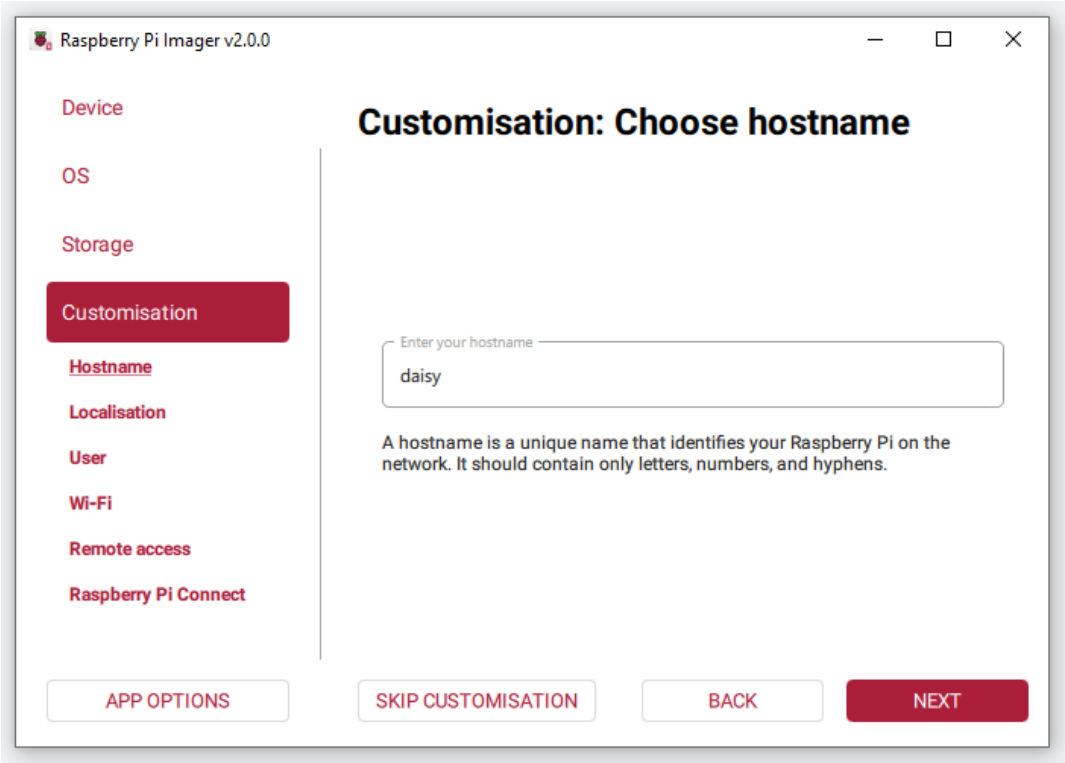


OS Customization Settings

1. Continue to the customization step.
    - If you will connect a monitor, keyboard, and mouse directly to your Raspberry Pi, you may click **SKIP CUSTOMISATION**.
    - If you plan to set up the Raspberry Pi *headless* (Wi-Fi remote access), you must complete the customization settings.

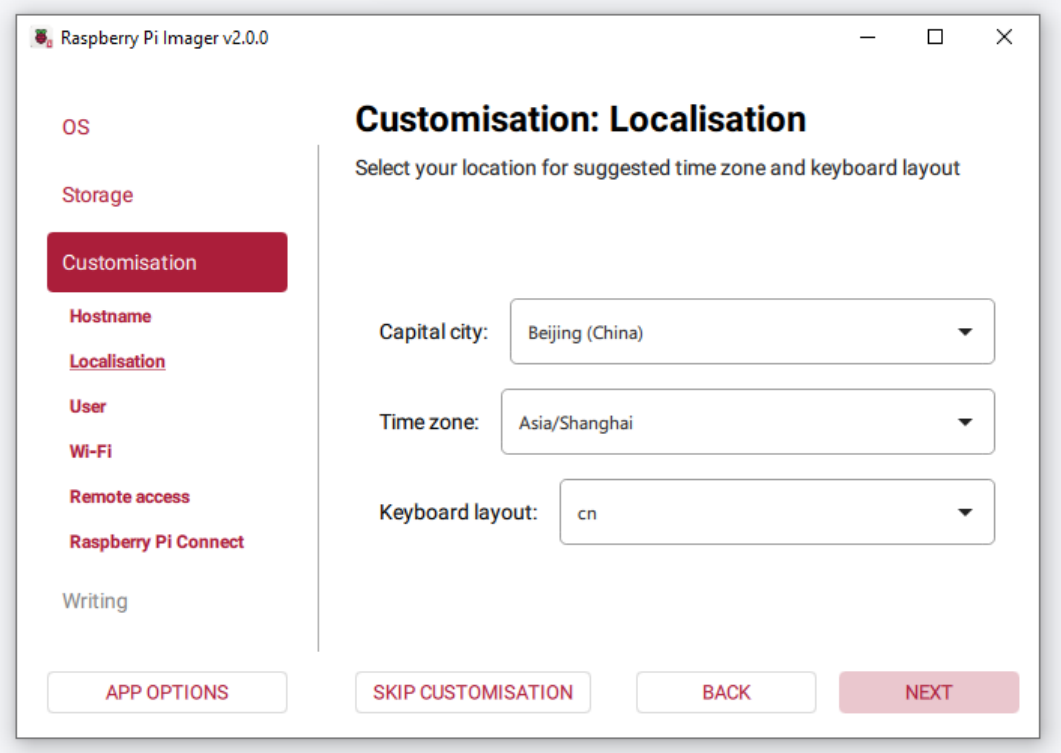


2. **Set Hostname**
   - Give your Raspberry Pi a unique hostname.
   - You can connect to it later using `hostname.local`.



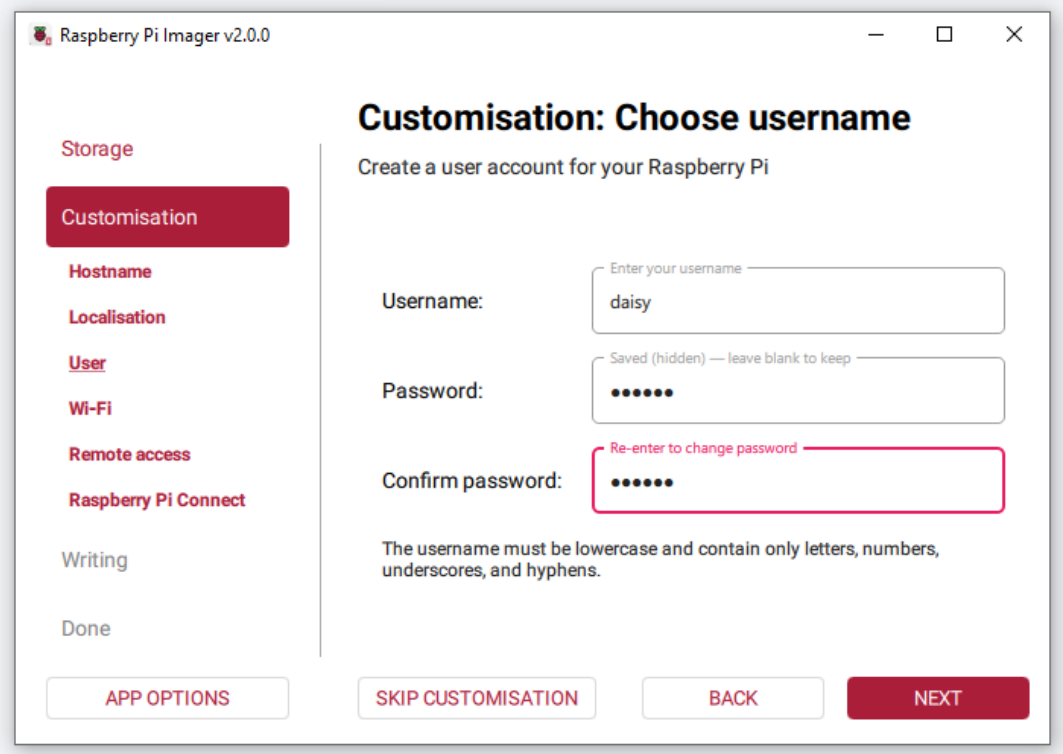
3. **Set Localisation**

   - Choose your capital city.
   - Imager will auto-complete the time zone and keyboard layout based on your selection, though you can adjust them if needed. Select Next.

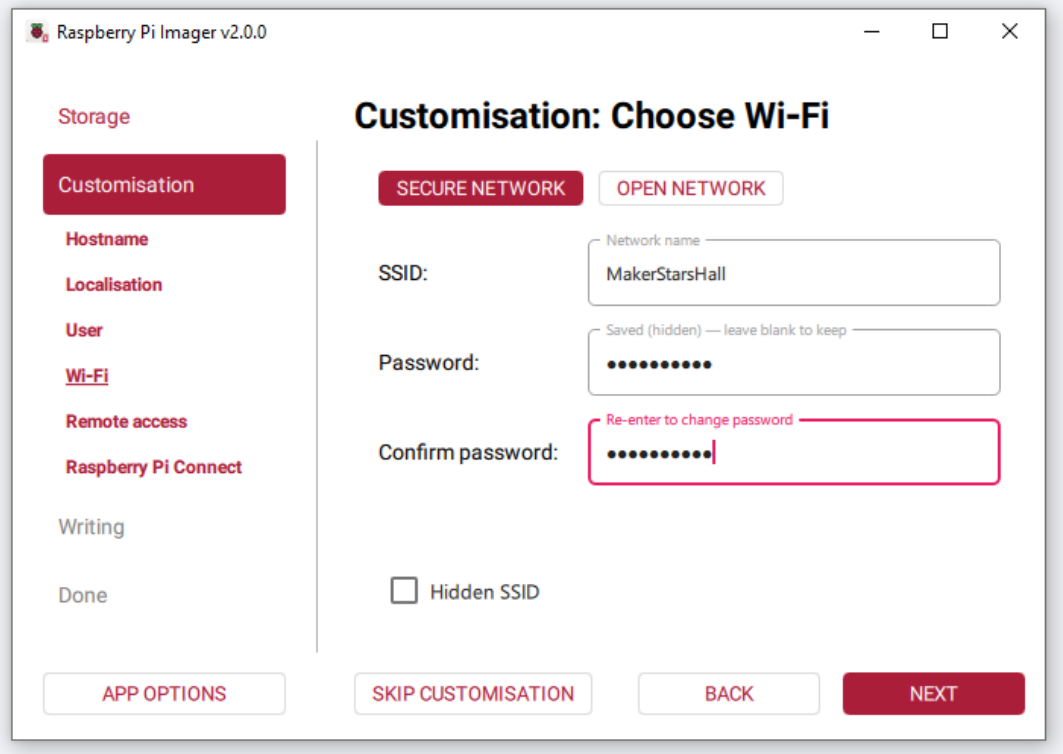


4. **Set Username & Password**

   Create a user account for your Raspberry Pi.

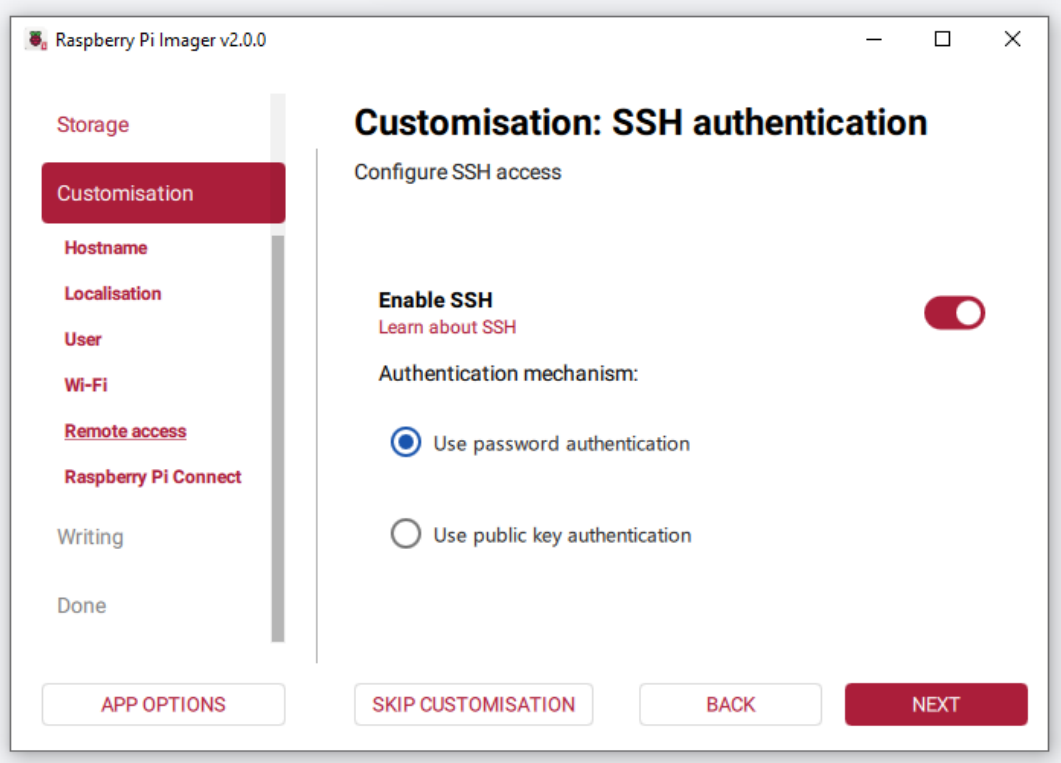


5. **Configure Wi-Fi**
   - Enter your Wi-Fi **SSID** (network name) and **password**.
   - Your Raspberry Pi will automatically connect on first boot.



6. **Enable SSH (Optional but Recommended)**

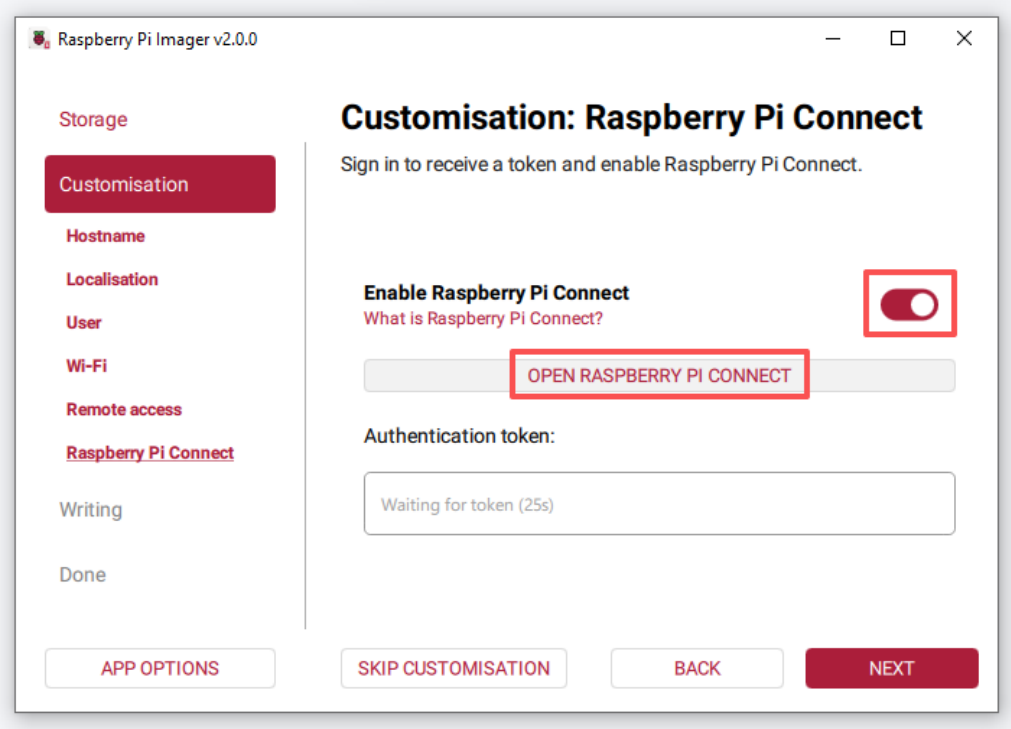
- Enabling SSH allows you to remotely log in from your computer.
- You may log in using your username/password or configure SSH keys.



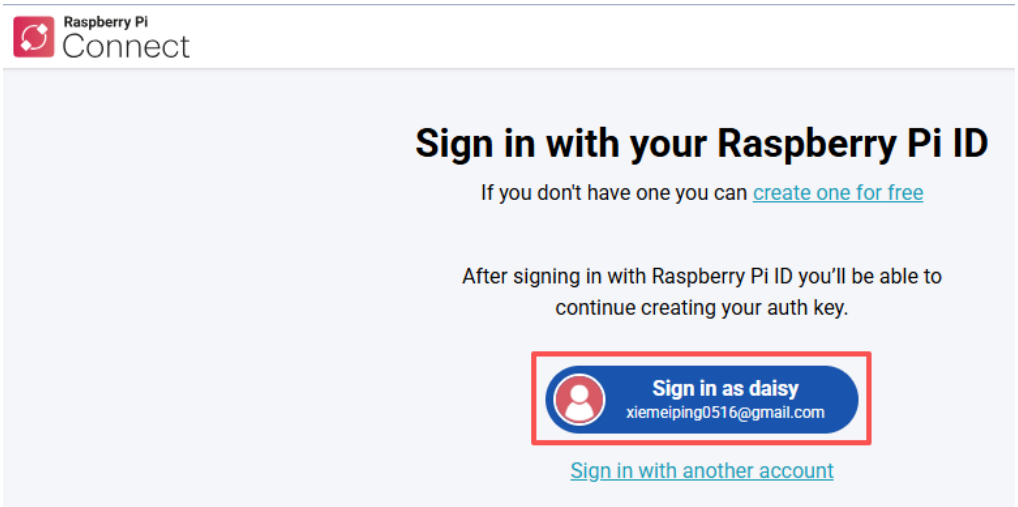
1. **Enable Raspberry Pi Connect (Optional)**

   Raspberry Pi Connect allows you to access your Raspberry Pi desktop from a web browser.

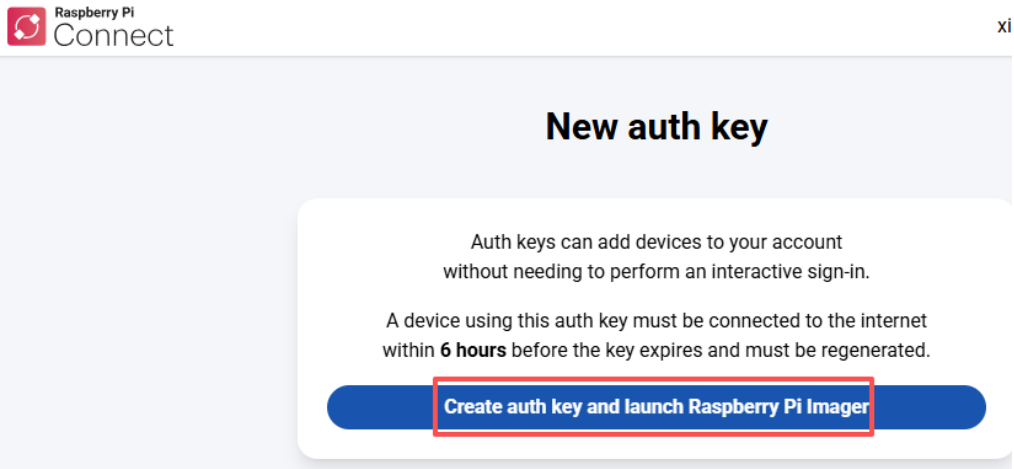
   - Turn on **Raspberry Pi Connect**, then click **OPEN RASPBERRY PI CONNECT**.



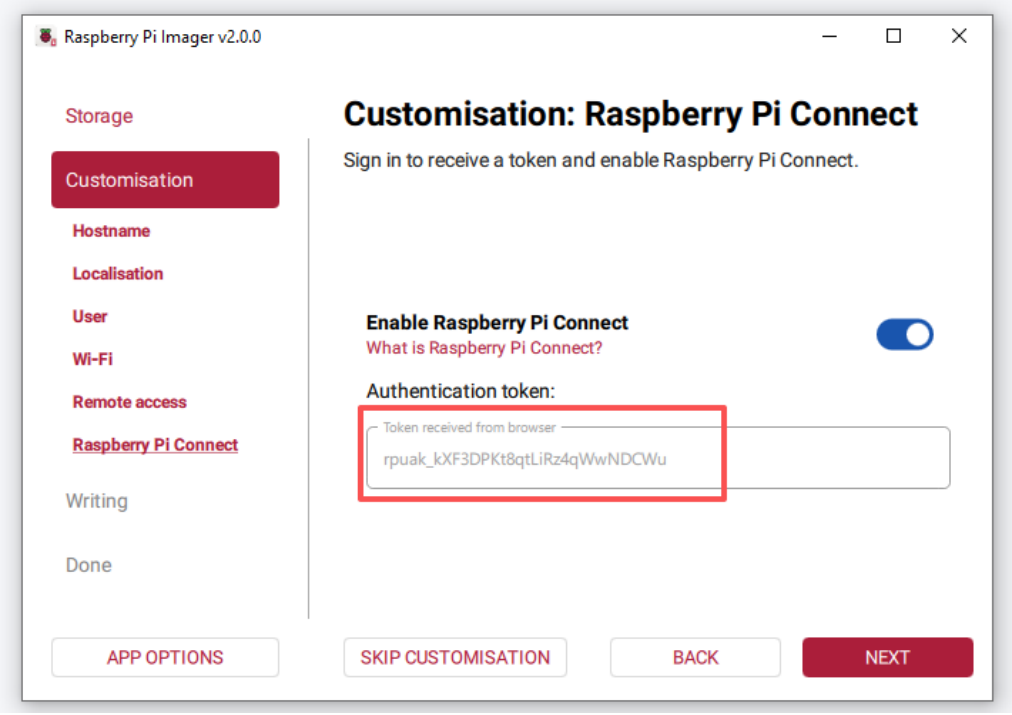
- The Raspberry Pi Connect website will open in your default browser. Log in to your Raspberry Pi ID account, or sign up if you don't have one yet.



- On the **New auth key** page, create your one-time auth key.
  - If your Raspberry Pi ID account isn't part of any organisation, select **Create auth key and launch Raspberry Pi Imager**.
  - If you belong to one or more organisations, choose one, then create the key and launch Imager.
  - Make sure to power on your Raspberry Pi and connect it to the internet before the key expires.

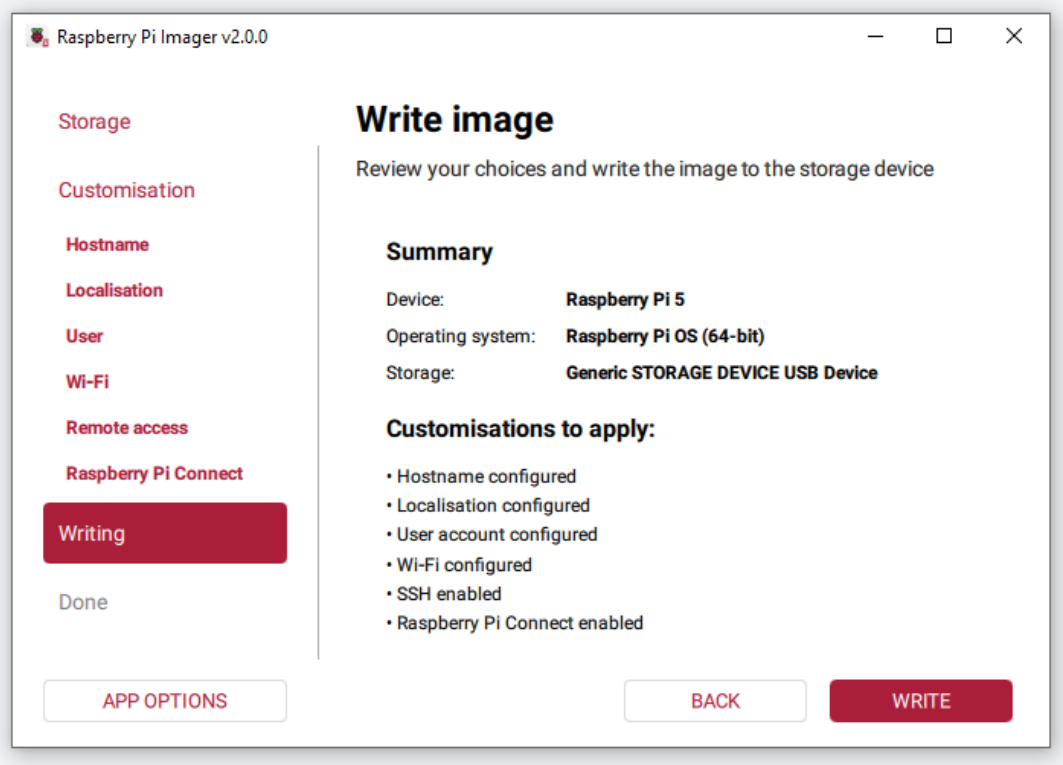


- Your browser may ask to open Raspberry Pi Imager — allow it.
  - Imager will open on the Raspberry Pi Connect tab, showing the authentication token.
  - If the token doesn't transfer automatically, open the **Having trouble?** section on the Raspberry Pi Connect page, copy the token, and paste it into Imager manually.

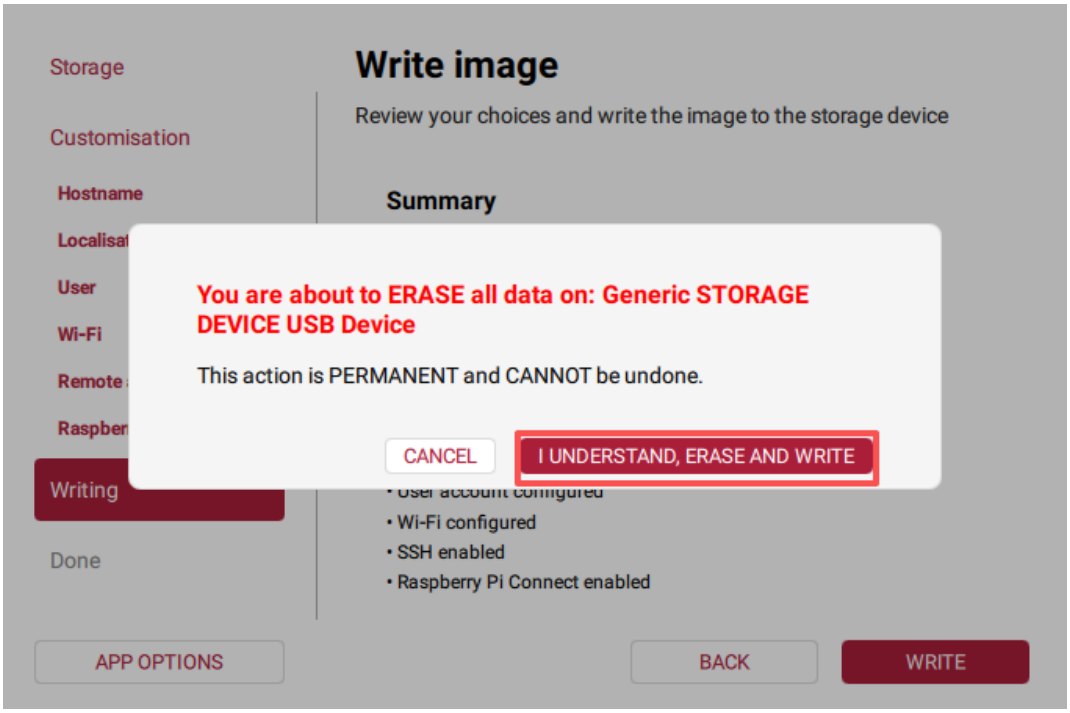


Write the OS Image

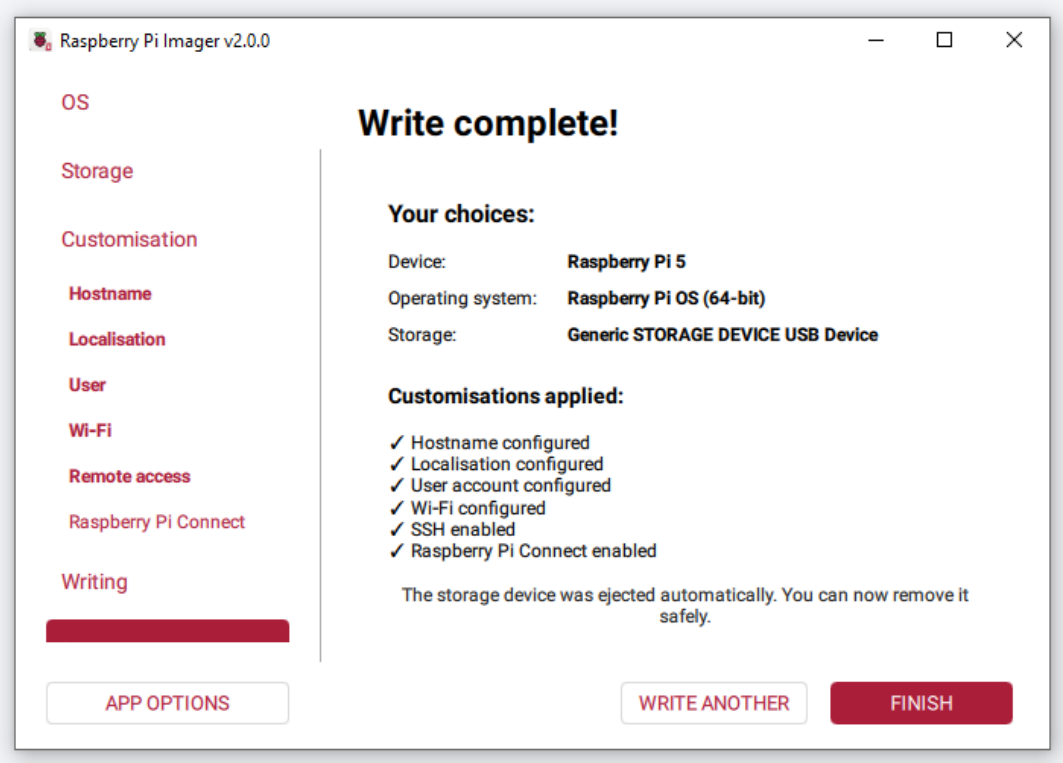
1. Review all settings and click **WRITE**.



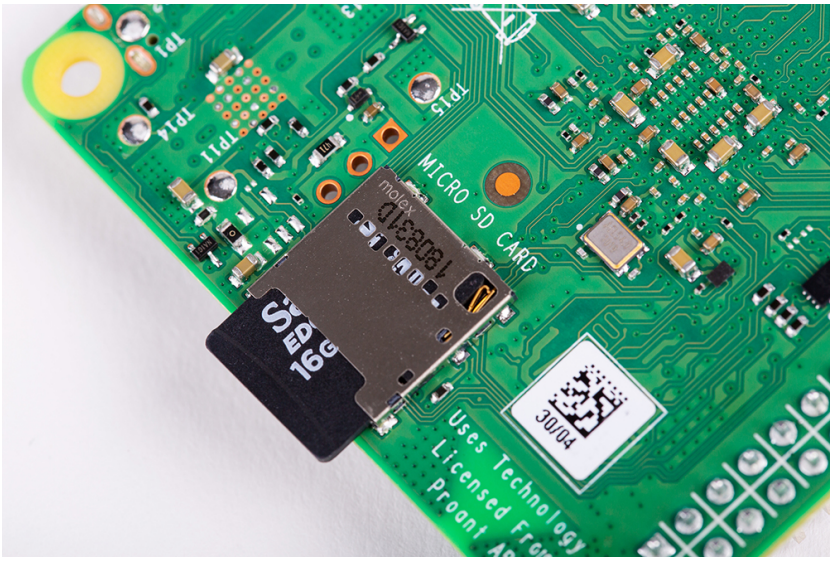
2. If the card already contains data, Raspberry Pi Imager will show a warning that all data on the device will be erased. Double-check that you selected the correct drive, then click **I UNDERSTAND, ERASE AND WRITE** to continue.



3. Wait for the writing and verification to finish. When it is done, Raspberry Pi Imager will show **Write complete!** and a summary of your choices. The storage device will be ejected automatically so you can remove it safely.



4. Remove the microSD card and insert it into the slot on the underside of your Raspberry Pi. Your Raspberry Pi is now ready to boot with the new OS!



Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.

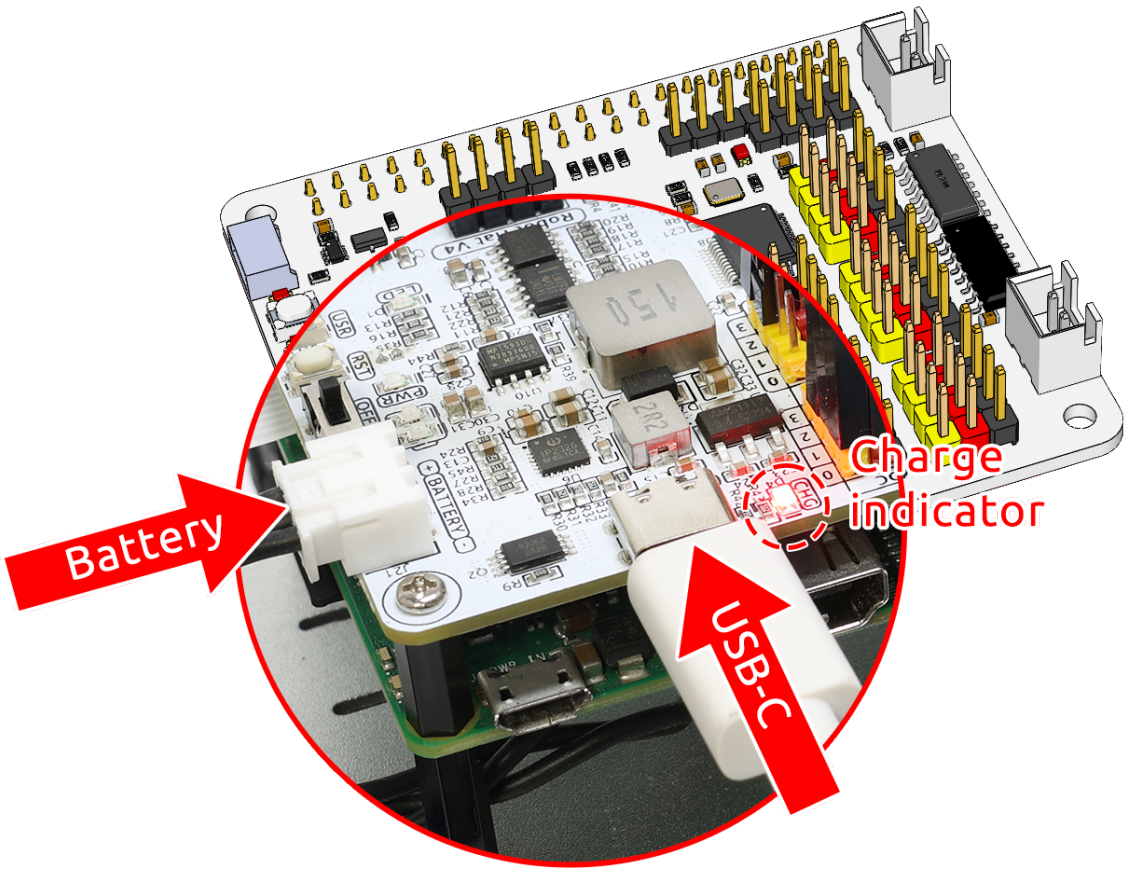
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

2.1.3 Power Supply for Raspberry Pi (Important)

Charge

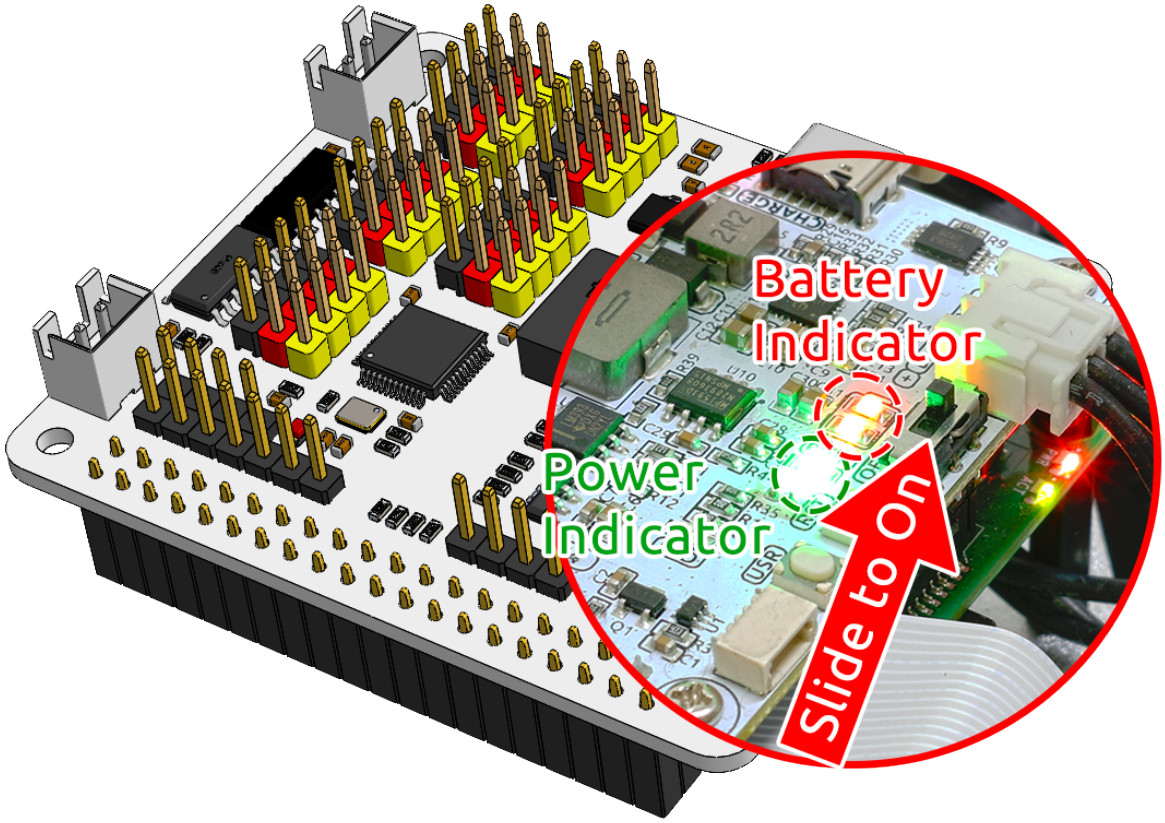
Insert the battery cable. Next, insert the USB-C cable to charge the battery. You will need to provide your own charger; we recommend a 5V 3A charger, or your commonly used smartphone charger will suffice.



Note: Connect an external Type-C power source to the Type-C port on the robot hat; it will immediately start charging the battery, and a red indicator light will illuminate.When the battery is fully charged, the red light will automatically turn off.

Power ON

Turn on the power switch. The Power indicator light and the battery level indicator light will illuminate.



Wait for a few seconds, and you will hear a slight beep, indicating that the Raspberry Pi has successfully booted.

Note: If both battery level indicator lights are off, please charge the battery. When you need extended programming or debugging sessions, you can keep the Raspberry Pi operational by inserting the USB-C cable to charge the battery simultaneously.

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

2.1.4 Set Up Your Raspberry Pi

To begin programming and controlling your Raspberry Pi, you first need to access it. This guide describes two common methods:

- Using a monitor, keyboard, and mouse
- Setting up a headless (no-screen) connection for remote access

---

Note: The Raspberry Pi Zero 2W installed on the robot is not easy to connect to a screen. We recommend using the **headless setup** method.

If You Have a Screen

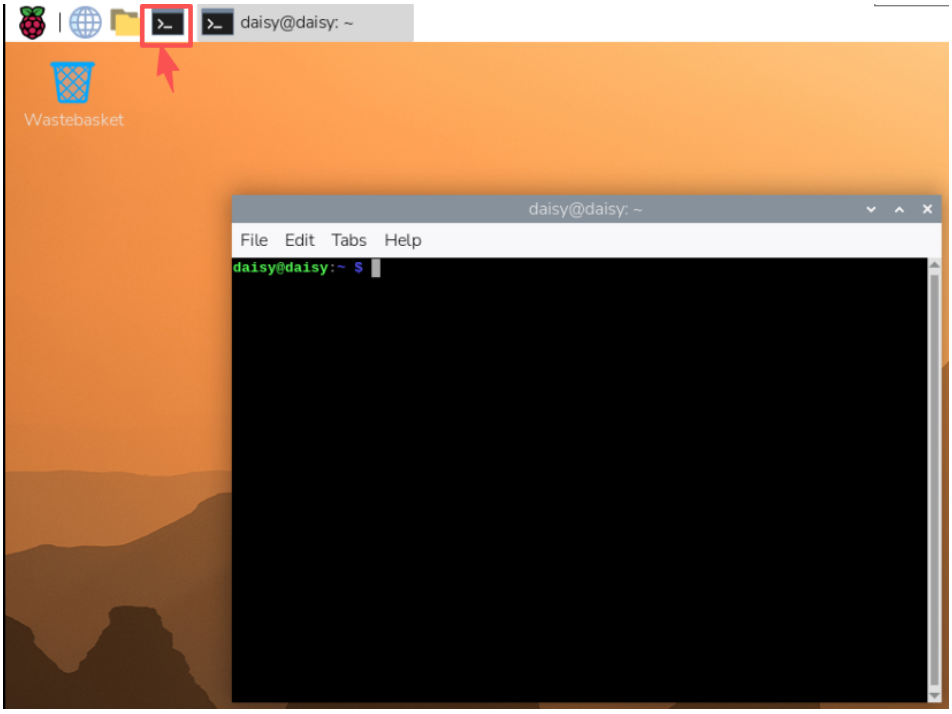
Required Components

- Raspberry Pi
- Official Power Supply
- MicroSD Card
- HDMI Cable (For Raspberry Pi 4/5, use **HDMI0**, the port nearest the power connector.)
- Monitor
- Keyboard and Mouse

Steps

1. Insert the microSD card into your Raspberry Pi.
2. Connect the keyboard, mouse, and monitor.
3. Power on your Raspberry Pi.
4. After booting, the Raspberry Pi OS desktop will appear.

5. Open a **Terminal** to enter commands.



If You Have No Screen (Headless)

Without a monitor, you can configure and log in to your Raspberry Pi remotely. This is the most convenient method for most users.

Required Components

- Raspberry Pi
- Official Power Supply
- MicroSD Card
- A computer on the same network

Tips

- Make sure you have completed all settings described in *OS Customization Settings* when installing the system with Raspberry Pi Imager.
- Ensure that your Raspberry Pi and your computer are on the same local network.
- For best stability, use Ethernet if available.

Connect via SSH

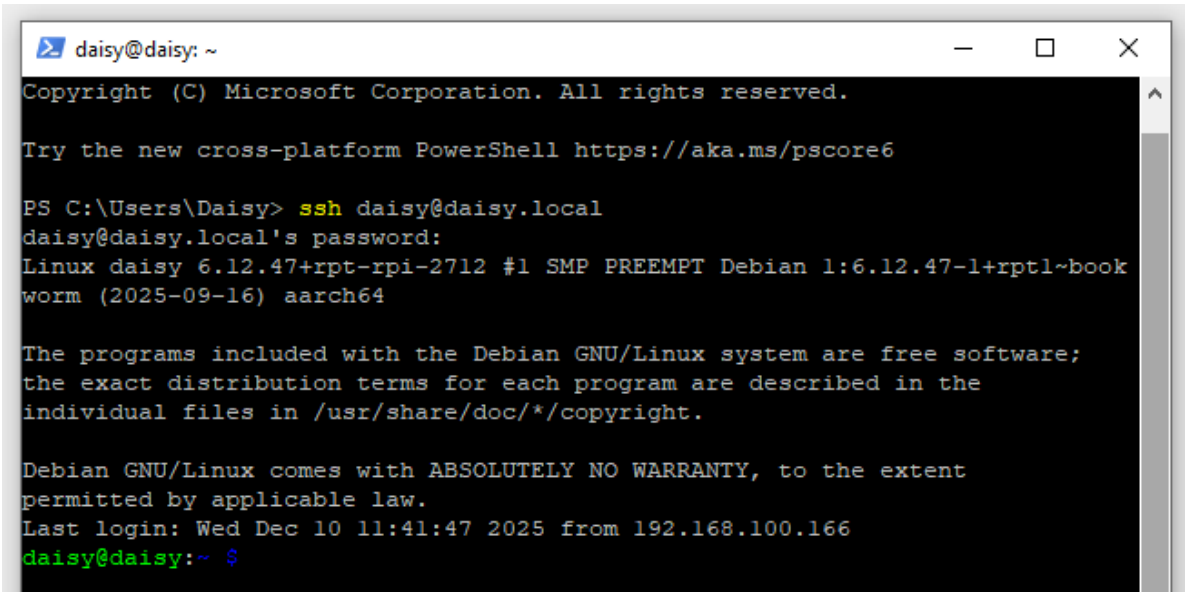
1. Open a terminal on your computer (Windows: **PowerShell**; macOS/Linux: **Terminal**) and connect to your Raspberry Pi:

```
ssh <username>@<hostname>.local
# Example:
ssh daisy@pi.local
```

2. Alternatively, locate your Pi's IP address from your router's DHCP list and connect with:

```
ssh <username>@<IP address>
# Example:
ssh daisy@192.168.1.42
```

3. On first login, type `yes` to confirm the SSH certificate.
4. Enter the password you configured in Raspberry Pi Imager. (Nothing appears while typing—this is normal.)
5. After login, you now have full command-line access.



Troubleshooting

- **ssh: Could not resolve hostname …**
  - Make sure the hostname is correct.
  - Try connecting using the Pi's IP address.
- **The term 'ssh' is not recognized… (Windows)**
  - OpenSSH is not installed. Install it manually or use a third-party SSH client.
  - See *Install OpenSSH via PowerShell* or login_windows.
- **Permission denied (publickey,password)**
  - Ensure you are using the username and password created in Raspberry Pi Imager.
- **Connection refused**
  - Wait 1–2 minutes after powering on.
  - Confirm that SSH was enabled in Raspberry Pi Imager.

Graphical Remote Access Options

If you prefer a graphical interface:

- *Remote Desktop*: Enable **VNC (Virtual Network Computing)** for a full desktop experience on your Pi.
- : Use Raspberry Pi Connect for secure remote access from anywhere, directly in a browser.

Now you can control your Raspberry Pi without a monitor, either through SSH for command-line operations, or with VNC / Raspberry Pi Connect for a graphical desktop experience.

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

2.1.5 Install All the Modules(Important)

Note: Python3 related packages must be installed if you are installing the Lite version OS.

```
sudo apt install git python3-pip python3-setuptools python3-smbus
```

1. Install `robot-hat` module.

   ```
   cd ~/
   git clone -b 2.5.x https://github.com/sunfounder/robot-hat.git --depth 1
   cd robot-hat
   sudo python3 install.py
   ```

2. Install `vilib` module.

   ```
   cd ~/
   git clone https://github.com/sunfounder/vilib.git
   cd vilib
   sudo python3 install.py
   ```

3. Install `pidog` module.

   ```
   cd ~/
   git clone https://github.com/sunfounder/pidog.git --depth 1
   cd pidog
   sudo pip3 install . --break
   ```

   This step will take a little time, so please be patient.

4. Run the script `i2samp.sh`.

   Finally, you need to run the script `i2samp.sh` to install the components required by the i2s amplifier, otherwise the robot will have no sound.

   ```
   cd ~/robot-hat
   sudo bash i2samp.sh
   ```

   Type y and press Enter three times to continue the script, start /dev/zero in the background, and then restart the machine.

Note: If there is no sound after restarting, you may need to run the `i2samp.sh` script multiple times.

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

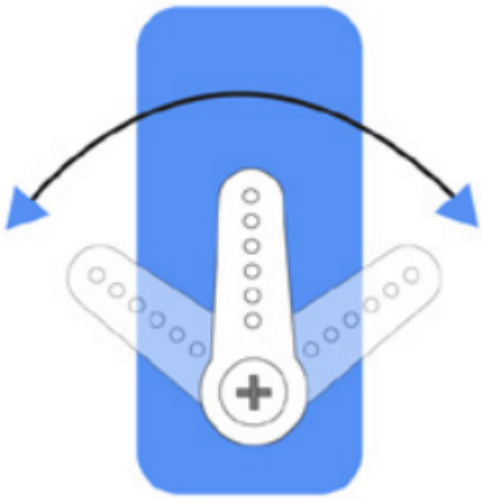
2.1.6 Servo Adjust(Important)

Warning: If your PiDog is V2 version, please skip this section, because the servo angles are already adjusted during assembly.

The angle range of the servo is -90~90, but the angle set at the factory is random, maybe 0°, maybe 45°; if we assemble it with such an angle directly, it will lead to a chaotic state after the robot runs the code, or worse, it will cause the servo to block and burn out.

So here we need to set all the servo angles to 0° and then install them, so that the servo angle is in the middle, no matter which direction to turn.

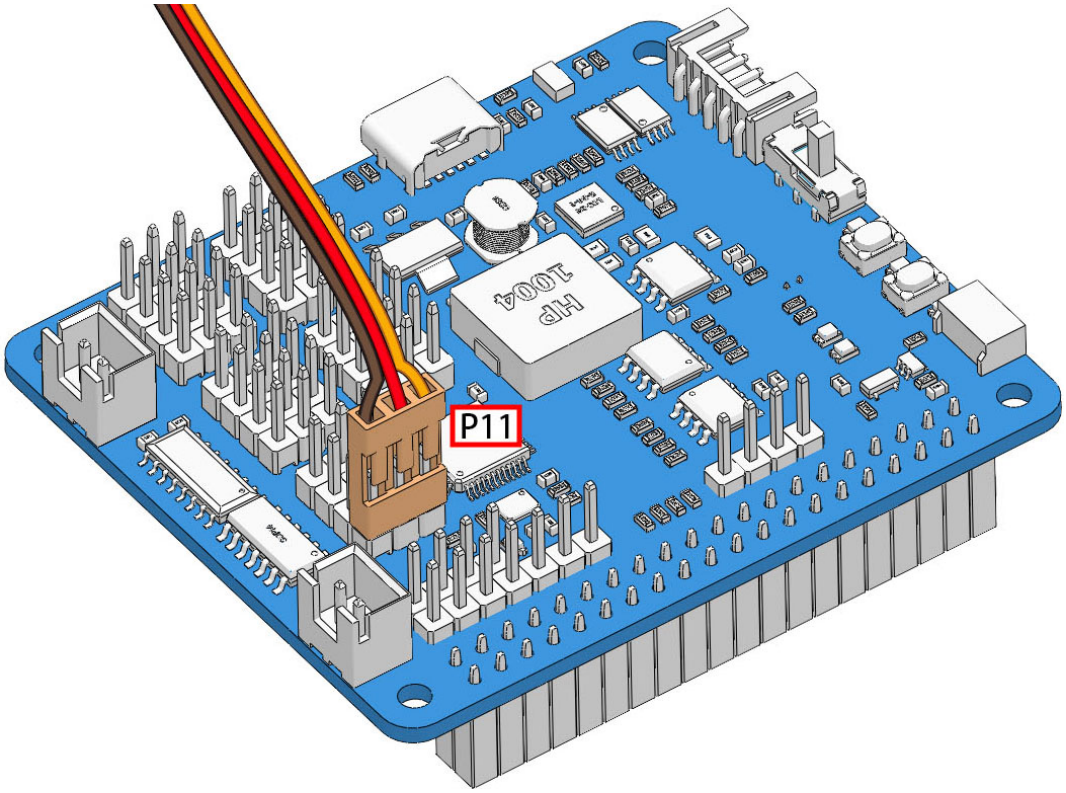
1. To ensure that the servo has been properly set to 0°, first insert the servo arm into the servo shaft and then gently rotate the rocker arm to a different angle. This servo arm is just to allow you to clearly see that the servo is rotating.



2. Now, run `servo_zeroing.py` in the `examples/` folder.

```
cd ~/pidog/examples
sudo python3 servo_zeroing.py
```

3. Next, plug the servo cable into the P11 port as follows, at the same time you will see the servo arm rotate to a position(This is the 0° position, which is a random location and may not be vertical or parallel.).



4. Now, remove the servo arm, ensuring the servo wire remains connected, and do not turn off the power. Then continue the assembly following the paper instructions.

Note:

- Do not unplug this servo cable before fixing it with the servo screw, you can unplug it after fixing it.
- Do not rotate the servo while it is powered on to avoid damage; if the servo shaft is not inserted at the right angle, pull the servo out and reinsert it.
- Before assembling each servo, you need to plug the servo cable into PWM pin and turn on the power to set its angle to 0°.

Video

In our assembly video from **3:40 to 7:23**, there is also a detailed tutorial for this chapter. You can follow the video instructions directly.

As soon as the assembly is completed, you need to calibrate the PiDog to prevent it from damaging the servo if there is a slight deviation in the assembly.

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

2.2 2. Calibrating PiDog

Introduction

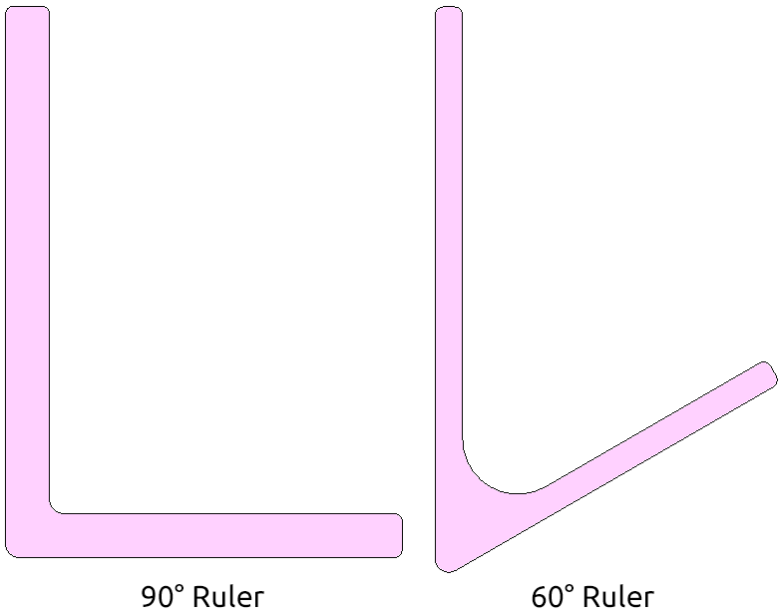
Calibrating your PiDog is a crucial step to ensure its stable and efficient operation. This process helps correct any imbalance or inaccuracy caused by assembly or structural errors. Follow the steps below carefully to make sure your PiDog walks smoothly and performs as expected.

If the deviation angle is too large, return to *Servo Adjust(Important)*, set the servo angle to 0°, and then reassemble PiDog according to the instructions.

Calibration Video

For a detailed guide, refer to the full calibration tutorial video. It will visually and step by step demonstrate how to accurately calibrate your PiDog.

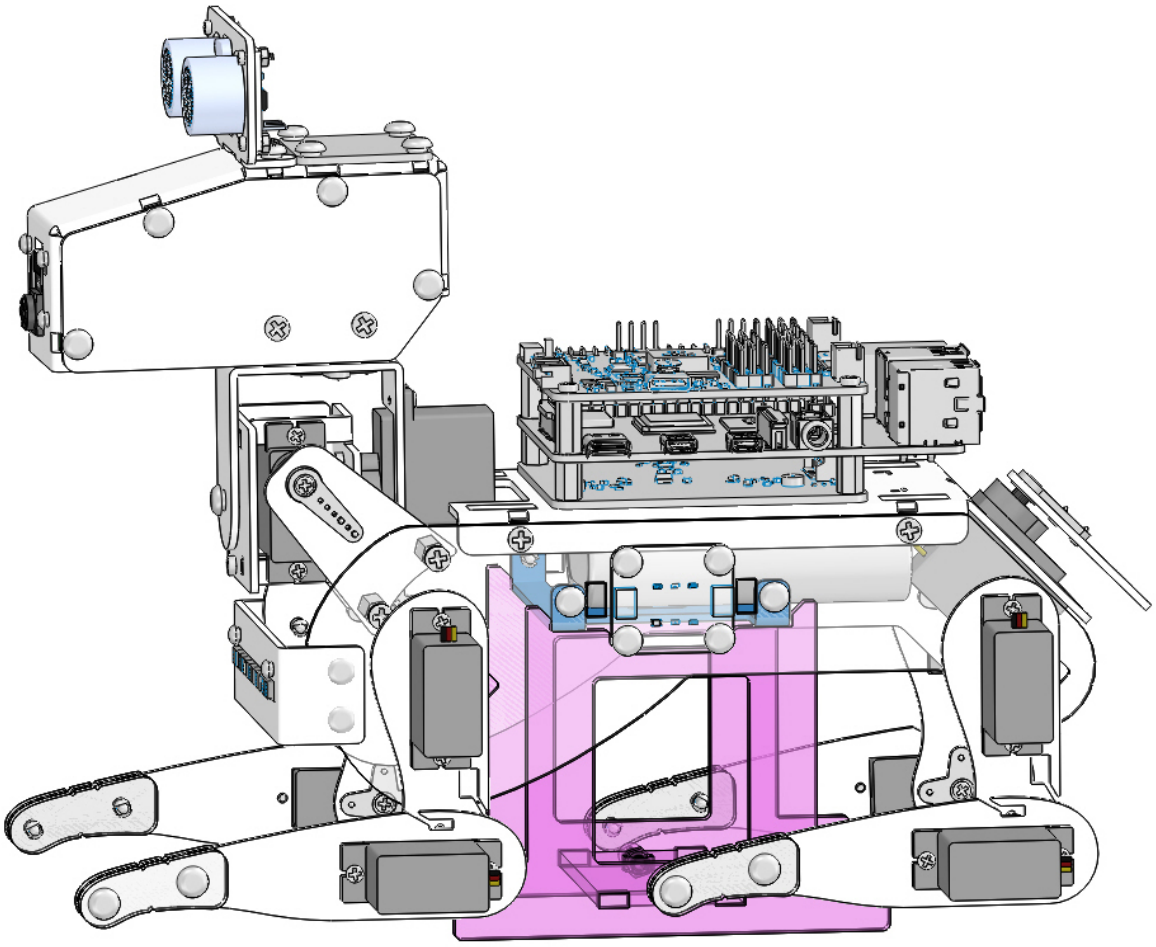
Note: The PiDog kit includes either a 90° or 60° calibration ruler. The video uses the 90° ruler, but the 60° calibration process is very similar. You can also refer to the illustrated step-by-step guide below.



Steps

Follow these steps:

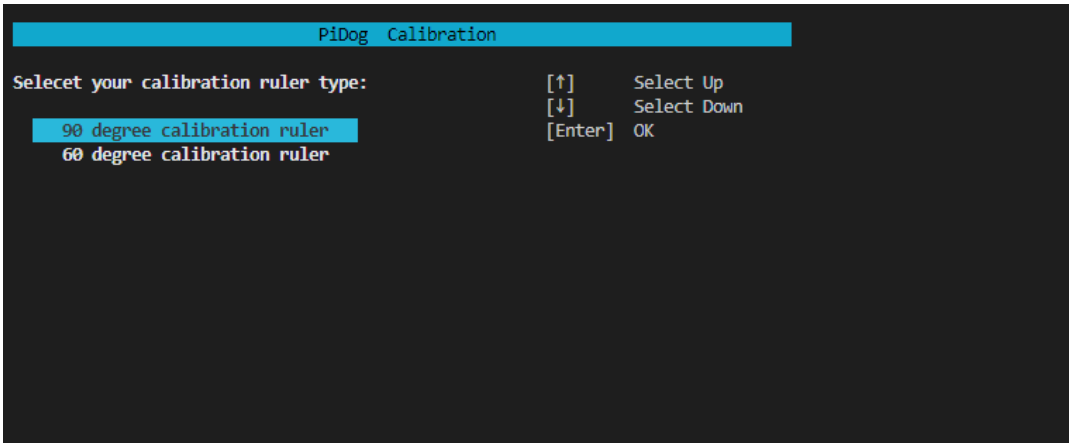
1. Place the PiDog on a flat platform.



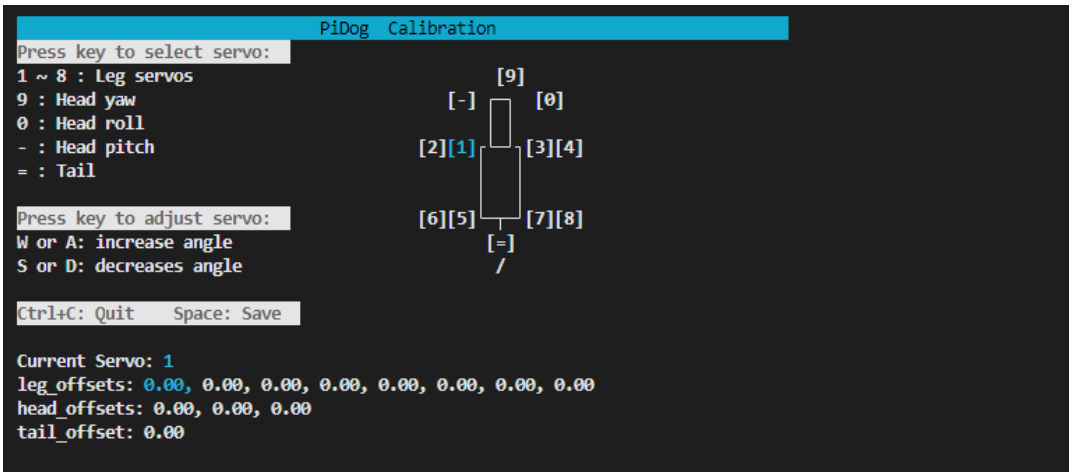
2. Navigate to the PiDog example code directory and run the `0_calibration.py` script.

```
cd ~/pidog/examples
sudo python3 0_calibration.py
```

3. After running the script, an interactive interface will appear in the terminal. Choose the type of calibration ruler you have: select option 1 for 90°, or option 2 for 60°.

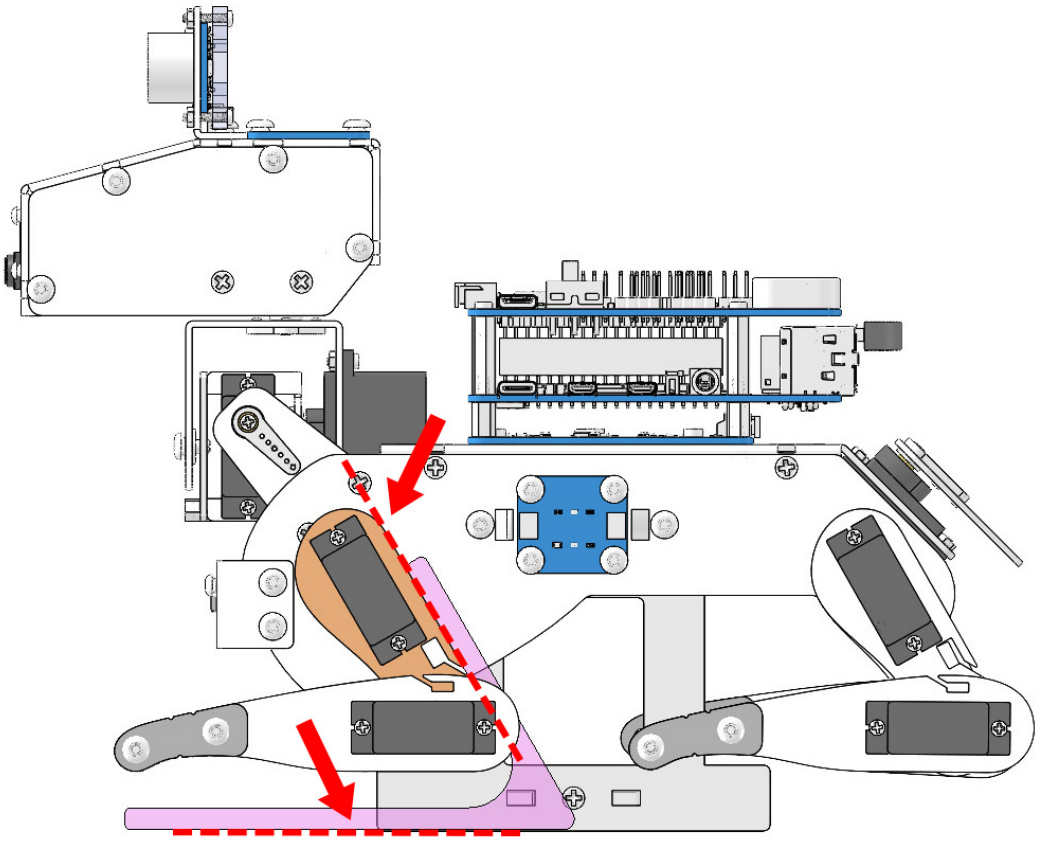


4. After making your selection, the following calibration interface will appear:

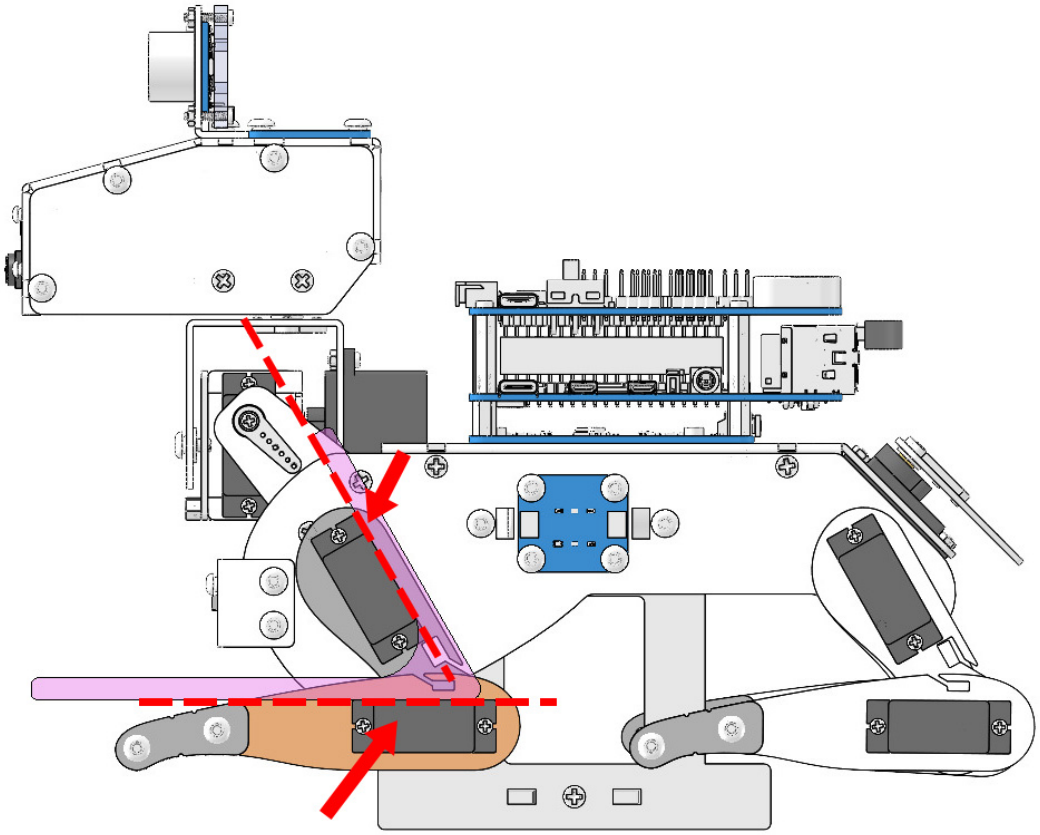


If you are using the 60° calibration ruler

1. Position the **calibration ruler (acrylic C-board)** as shown, with the long edge on the horizontal surface. Press 1 in the terminal and use the `w` and `s` keys to adjust the alignment.



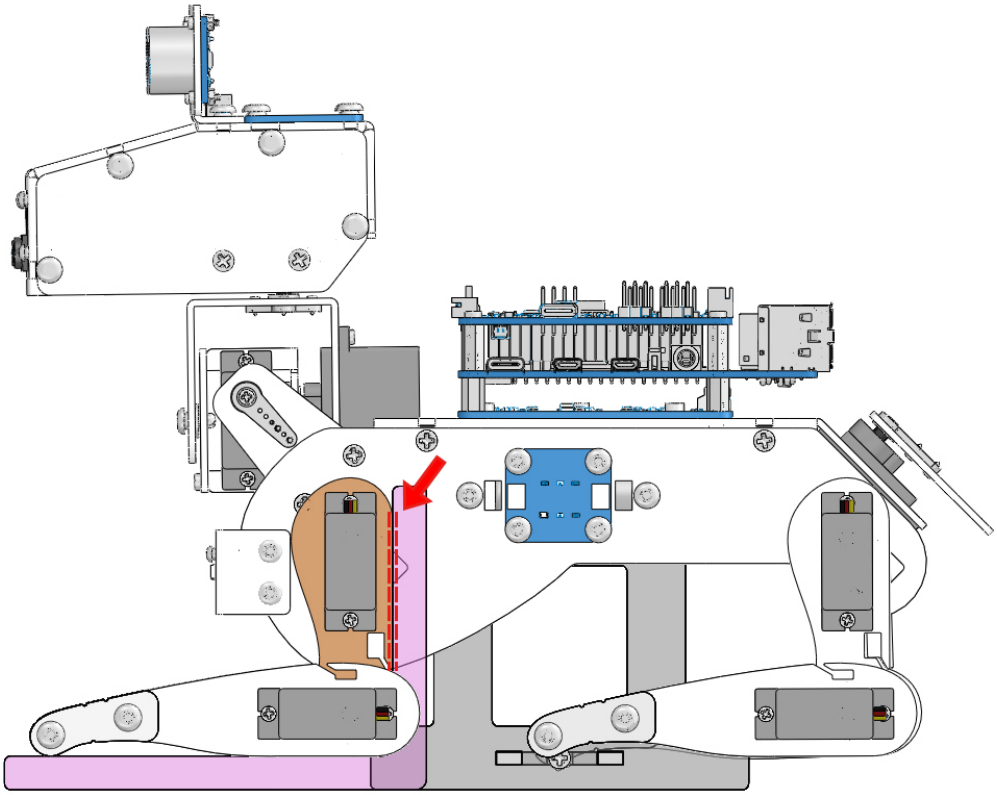
2. Reposition the **calibration ruler** as shown in the figure below. Press 2 in the terminal and fine-tune the alignment using the `w` and `s` keys.



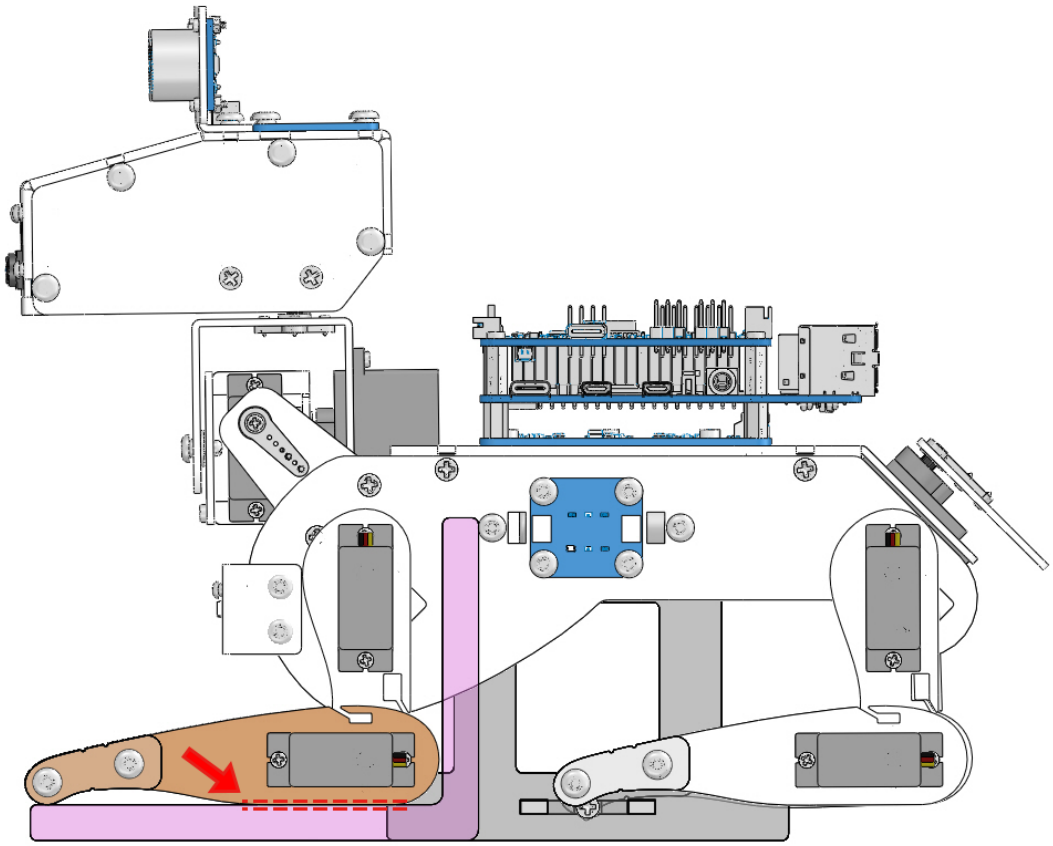
3. Repeat the calibration process for servos 3 to 8 to ensure all four legs of the PiDog are properly calibrated.

If you are using the 90° calibration ruler

1. Position the **calibration ruler (acrylic C-board)** as shown. Press `1` in the terminal, then use `w` and `s` to align the edges with the illustration.



2. Reposition the **calibration ruler** (**acrylic C-board**) as shown. Press `2` in the terminal, then adjust again with `w` and `s`.



3. Repeat the calibration procedure for servos 3 to 8 to ensure all four legs of the PiDog are properly calibrated.

Completing the Calibration

- Once all servos are calibrated, rerun the PiDog walking or posture example code to check if the movements are smooth.
- If there is still deviation, return to the calibration program for fine-tuning.
- It is highly recommended to complete this step after your initial assembly to ensure stable performance during operation.

Tip: To avoid recalibrating in the future, you can record the servo angles or export the configuration file after calibration. This makes it easy to restore settings quickly next time.

You can also have PiDog achieve the following project effects.

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.

- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

2.3 3. Fun Python Projects

Here, we delve into an exciting assortment of projects that showcase the versatility and capabilities of the PiDog. From the basics of setting a wakeup routine in *1. Wake Up* to the advanced dynamics of ball tracking in *13. Ball Track*, each project offers a unique glimpse into the world of Python programming for robotics. Whether you're keen on making your PiDog patrol an area, respond to commands, execute pushups, or even howl on command, there's a project tailored for you. Furthermore, for those looking to extend their PiDog's capabilities to computer interfaces, we have tutorials on keyboard and app control as well. Dive in, and embark on a journey of discovery and fun with these hands-on Python projects for your PiDog!

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

2.3.1 1. Wake Up

This is PiDog's first project. It will wake your PiDog from a deep sleep.

Run the Code

```
cd ~/pidog/examples
sudo python3 1_wake_up.py
```

After the code is executed, PiDog will perform the following actions in sequence:

Stretch, twist, sit, wag its tail, pant.

Code

Note: You can **Modify/Reset/Copy/Run/Stop** the code below. But before that, you need to go to source code path like `pidog\examples`. After modifying the code, you can run it directly to see the effect.

```
#!/usr/bin/env python3
from pidog import Pidog
from time import sleep
```

(continues on next page)

(continued from previous page)

```
from preset_actions import pant
from preset_actions import body_twisting

my_dog = Pidog(head_init_angles=[0, 0, -30])
sleep(1)

def wake_up():
    # stretch
    my_dog.rgb_strip.set_mode('listen', color='yellow', bps=0.6, brightness=0.8)
    my_dog.do_action('stretch', speed=50)
    my_dog.head_move([[0, 0, 30]]*2, immediately=True)
    my_dog.wait_all_done()
    sleep(0.2)
    body_twisting(my_dog)
    my_dog.wait_all_done()
    sleep(0.5)
    my_dog.head_move([[0, 0, -30]], immediately=True, speed=90)
    # sit and wag_tail
    my_dog.do_action('sit', speed=25)
    my_dog.wait_legs_done()
    my_dog.do_action('wag_tail', step_count=10, speed=100)
    my_dog.rgb_strip.set_mode('breath', color=[245, 10, 10], bps=2.5, brightness=0.8)
    pant(my_dog, pitch_comp=-30, volume=80)
    my_dog.wait_all_done()
    # hold
    my_dog.do_action('wag_tail', step_count=10, speed=30)
    my_dog.rgb_strip.set_mode('breath', 'pink', bps=0.5)
    while True:
        sleep(1)

if __name__ == "__main__":
    try:
        wake_up()
    except KeyboardInterrupt:
        pass
    except Exception as e:
        print(f"\033[31mERROR: {e}\033[m")
    finally:
        my_dog.close()
```

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

2.3.2 2. Function Demonstration

This project shows you all of PiDog's usual actions and sounds.

You can make PiDog make actions or make sounds by entering the serial number.

The motion/sound effects currently included in this example are listed below.

Actions:	Sound Effect:
1.stand	16.angry
2.sit	17.confused_1
3.lie	18.confused_2
4.lie_with_hands_out	19.confused_3
5.trot	20.growl_1
6.forward	21.growl_2
7.backward	22.howling
8.turn_left	23.pant
9.turn_right	24.single_bark_1
10.doze_off	25.single_bark_2
11.stretch	26.snoring
12.pushup	27.woohoo
13.shake_head	
14.tilting_head	
15.wag_tail	

Run the Code

```
cd ~/pidog/examples
sudo python3 2_function_demonstration.py
```

After running this example, you input `1` and press `ENTER`, PiDog will stand; input `2`, PiDog will sit down; input `27`, PiDog will issue "woohoo~ ".

Press `Ctrl+C` to exit the program.

Code

Note: You can **Modify/Reset/Copy/Run/Stop** the code below. But before that, you need to go to source code path like `pidog\examples`. After modifying the code, you can run it directly to see the effect.

```
#!/usr/bin/env python3
from time import sleep
from pidog import Pidog
import os
import curses
import curses_utils

# init pidog
```

(continues on next page)

(continued from previous page)

```
# ====================================
my_dog = Pidog()
sleep(0.5)

# global variables
# ====================================
actions = [
    # name, head_pitch_adjust(-1, use last_pitch), speed
    ['stand', 0, 50],
    ['sit', -30, 50],
    ['lie', 0, 20],
    ['lie_with_hands_out', 0,  20],
    ['trot', 0, 95],
    ['forward', 0, 98],
    ['backward', 0, 98],
    ['turn_left', 0, 98],
    ['turn_right', 0, 98],
    ['doze_off', -30, 90],
    ['stretch', 20, 20],
    ['push_up', -30, 50],
    ['shake_head', -1, 90],
    ['tilting_head', -1, 60],
    ['wag_tail', -1, 100],
]
actions_len = len(actions)

sound_effects = []
# change working directory
abspath = os.path.abspath(os.path.dirname(__file__))
# print(abspath)
os.chdir(abspath)
for name in os.listdir('../sounds'):
    sound_effects.append(name.split('.')[0])
sound_effects.sort()
sound_len = len(sound_effects)
# limit sound quantity
if sound_len > actions_len:
    sound_len = actions_len
    sound_effects = sound_effects[:actions_len]

last_index = 0
last_display_index = 0
exit_flag = False
last_head_pitch = 0

STANDUP_ACTIONS = ['trot', 'forward', 'backward', 'turn_left', 'turn_right']

# define pad size
# ====================================
curses_utils.PAD_Y = 22
curses_utils.PAD_X = 70
```

(continues on next page)

(continued from previous page)

```
# display fuctions
# ====================================
def display_head(subpad):
    title = "Function Demonstration"
    tip1 = "Input Function number to see how it goes."
    tip2 = "Actions will repeat 10 times."
    type_name_1 = "Actions:"
    type_name_2 = "Sound Effect:"
    tip3 = "(need to run with sudo)"

    curses_utils.clear_line(subpad, 0, color=curses_utils.BLACK_BLUE)
    subpad.addstr(0, 2, title, curses_utils.BLACK_BLUE | curses.A_BOLD)
    subpad.addstr(1, 2, tip1, curses_utils.GRAY)
    subpad.addstr(2, 2, tip2, curses_utils.GRAY)
    curses_utils.clear_line(subpad, 3, color=curses_utils.WHITE_GRAY)
    subpad.addstr(3, 2, type_name_1, curses_utils.WHITE_GRAY)
    subpad.addstr(3, 30, type_name_2, curses_utils.WHITE_GRAY)
    subpad.addstr(3, 31+len(type_name_2), tip3, curses_utils.YELLOW_GRAY)

def display_selection(subpad, index):
    global last_display_index
    # reset last selection
    if last_display_index > actions_len + sound_len-1 or last_display_index < 0:
        last_display_index = 0
    if last_display_index != index:
        if last_display_index < actions_len:
            subpad.addstr(last_display_index, 2, f"{last_display_index+1}. {actions[last_
↪display_index][0]}", curses_utils.LIGHT_GRAY)
        else:
            sound_index = last_display_index-actions_len
            subpad.addstr(sound_index, 30, f"{last_display_index+1}. {sound_
↪effects[sound_index]}", curses_utils.LIGHT_GRAY)
        last_display_index = index
    # highlight currernt selection
    if index > actions_len + sound_len-1 or index < 0:
        pass
    elif index < actions_len:
        subpad.addstr(index, 2, f"{index+1}. {actions[index][0]}", curses_utils.WHITE_
↪BLUE)
    else:
        sound_index = index-actions_len
        subpad.addstr(sound_index, 30, f"{index+1}. {sound_effects[sound_index]}",␣
↪curses_utils.WHITE_BLUE)

def display_actions(subpad):
    for i in range(actions_len):
        subpad.addstr(i, 2, f"{i+1}. {actions[i][0]}", curses_utils.LIGHT_GRAY)
    for i in range(sound_len):
        subpad.addstr(i, 30, f"{i+actions_len+1}. {sound_effects[i]}", curses_utils.
↪LIGHT_GRAY)

def display_bottom(subpad):
```

(continues on next page)

(continued from previous page)

```
    curses_utils.clear_line(subpad, 0, color=curses_utils.WHITE_GRAY)
    subpad.addstr(0, 0, "Enter function number: ", curses_utils.WHITE_GRAY)
    subpad.addstr(0, curses_utils.PAD_X-16, "Ctrl^C to quit", curses_utils.WHITE_GRAY)


def do_function(index):
    global last_index, last_head_pitch
    my_dog.body_stop()
    if index < 0:
        return
    if index < actions_len:
        name, head_pitch_adjust, speed = actions[index]
        # If last action is push_up, then lie down first
        if last_index < len(actions) and actions[last_index][0] in ('push_up'):
            last_head_pitch = 0
            my_dog.do_action('lie', speed=60)
        # If this action is trot, forward, turn left, turn right and backward, and, last␣
↪action is not, then stand up
        if name in STANDUP_ACTIONS and last_index < len(actions) and actions[last_
↪index][0] not in STANDUP_ACTIONS:
            last_head_pitch = 0
            my_dog.do_action('stand', speed=60)
        if head_pitch_adjust != -1:
            last_head_pitch = head_pitch_adjust
        my_dog.head_move_raw([[0, 0, last_head_pitch]], immediately=False, speed=60)
        my_dog.do_action(name, step_count=10, speed=speed, pitch_comp=last_head_pitch)
        last_index = index
    elif index < actions_len + sound_len:
        my_dog.speak(sound_effects[index - len(actions)], volume=80)
        last_index = index

def main(stdscr):
    # reset screen
    stdscr.clear()
    stdscr.move(4, 0)
    stdscr.refresh()

    # disable cursor
    curses.curs_set(0)

    # init color
    curses.start_color()
    curses.use_default_colors()
    curses_utils.init_preset_colors()
    curses_utils.init_preset__color_pairs()

    # init pad
    pad = curses.newpad(curses_utils.PAD_Y, curses_utils.PAD_X)

    # init subpad
    head_pad = pad.subpad(4, curses_utils.PAD_X, 0, 0)
    selection_pad = pad.subpad(actions_len, curses_utils.PAD_X, 4, 0)
```

(continues on next page)

(continued from previous page)

```
    bottom_pad = pad.subpad(1, curses_utils.PAD_X, actions_len+4, 0)
    # add content to a
    display_head(head_pad)
    display_actions(selection_pad)
    display_head(head_pad)
    curses_utils.pad_refresh(pad)
    curses_utils.pad_refresh(selection_pad)

    # for i in range(2):
    #     for i in range(30):
    #         display_selection(selection_pad, i)
    #         curses_utils.pad_refresh(selection_pad)
    #         sleep(0.1)

    # enable cursor and echo
    curses.curs_set(0)
    curses.echo()

    while True:
        # draw bottom bar
        display_bottom(bottom_pad)
        curses_utils.pad_refresh(bottom_pad)
        # reset cursor
        stdscr.move(actions_len+4, 23)
        stdscr.refresh()
        # red key
        key = stdscr.getstr()
        try:
            index = int(key) - 1
        except ValueError:
            index = -1
        # display selection
        display_selection(selection_pad, index)
        curses_utils.pad_refresh(selection_pad)
        # do fuction
        do_function(index)

        sleep(0.2)

if __name__ == "__main__":
    try:
        curses.wrapper(main)
    except KeyboardInterrupt:
        pass
    except Exception as e:
        print(f"\033[31mERROR: {e}\033[m")
    finally:
        my_dog.close()
```

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

2.3.3 3. Patrol

In this project, PiDog makes a vivid behavior: patrolling.

PiDog will walk forward, if there is an obstacle in front of it, it will stop and bark.

Run the Code

```
cd ~/pidog/examples
sudo python3 3_patrol.py
```

After running this example, PiDog will wag its tail, scan left and right, and walk forward.

Code

Note: You can **Modify/Reset/Copy/Run/Stop** the code below. But before that, you need to go to source code path like `pidog\examples`. After modifying the code, you can run it directly to see the effect.

```
#!/usr/bin/env python3
import time
from pidog import Pidog
from preset_actions import bark

t = time.time()
my_dog = Pidog()
my_dog.do_action('stand', speed=80)
my_dog.wait_all_done()
time.sleep(.5)

DANGER_DISTANCE = 15

stand = my_dog.legs_angle_calculation([[0, 80], [0, 80], [30, 75], [30, 75]])

def patrol():
    distance = round(my_dog.ultrasonic.read_distance(), 2)
    print(f"distance: {distance} cm", end="", flush=True)

    # danger
    if distance < DANGER_DISTANCE:
        print("\033[0;31m DANGER !\033[m")
```

(continues on next page)

(continued from previous page)

```
        my_dog.body_stop()
        head_yaw = my_dog.head_current_angles[0]
        # my_dog.rgb_strip.set_mode('boom', 'red', bps=2)
        my_dog.rgb_strip.set_mode('bark', 'red', bps=2)
        my_dog.tail_move([[0]], speed=80)
        my_dog.legs_move([stand], speed=70)
        my_dog.wait_all_done()
        time.sleep(0.5)
        bark(my_dog, [head_yaw, 0, 0])

        while distance < DANGER_DISTANCE:
            distance = round(my_dog.ultrasonic.read_distance(), 2)
            if distance < DANGER_DISTANCE:
                print(f"distance: {distance} cm \033[0;31m DANGER !\033[m")
            else:
                print(f"distance: {distance} cm", end="", flush=True)
            time.sleep(0.01)
    # safe
    else:
        print("")
        my_dog.rgb_strip.set_mode('breath', 'white', bps=0.5)
        my_dog.do_action('forward', step_count=2, speed=98)
        my_dog.do_action('shake_head', step_count=1, speed=80)
        my_dog.do_action('wag_tail', step_count=5, speed=99)


if __name__ == "__main__":
    try:
        while True:
            patrol()
            time.sleep(0.01)
    except KeyboardInterrupt:
        pass
    except Exception as e:
        print(f"\033[31mERROR: {e}\033[m")
    finally:
        my_dog.close()
```

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

2.3.4 4. Response

In this project, PiDog will interact with you in an interesting way.

If you reach out and grab PiDog's head from the front, it will bark vigilantly.

But if you reach out from behind it and pet its head, it will enjoy it very much.

Run the Code

```
cd ~/pidog/examples
sudo python3 4_response.py
```

After running this example, PiDog's ultrasonic module will detect whether there is an obstacle ahead, If it detects your hand, it makes the breathing light glow red, takes a step back, and barks.

At the same time, the touch sensor will also work. If the touch sensor is stroked (not just touched), PiDog will shake its head, wag its tail, and show a comfortable look.

Code

Note: You can **Modify/Reset/Copy/Run/Stop** the code below. But before that, you need to go to source code path like `pidog\examples`. After modifying the code, you can run it directly to see the effect.

```
#!/usr/bin/env python3
from pidog import Pidog
from time import sleep
from math import sin
from preset_actions import bark_action

my_dog = Pidog()
sleep(0.1)

def lean_forward():
    my_dog.speak('angry', volume=80)
    bark_action(my_dog)
    sleep(0.2)
    bark_action(my_dog)
    sleep(0.8)
    bark_action(my_dog)

def head_nod(step):
    y = 0
    r = 0
    p = 30
    angs = []
    for i in range(20):
        r = round(10*sin(i*0.314), 2)
        p = round(20*sin(i*0.314) + 10, 2)
        angs.append([y, r, p])

    my_dog.head_move(angs*step, immediately=False, speed=80)

def alert():
```

(continues on next page)

(continued from previous page)

```
    my_dog.do_action('stand', step_count=1, speed=90)
    my_dog.rgb_strip.set_mode('breath', color='pink', bps=1, brightness=0.8)
    while True:
        print(
            f'distance.value: {round(my_dog.ultrasonic.read_distance(), 2)} cm, touch
↪{my_dog.dual_touch.read()}')
        # alert
        if my_dog.ultrasonic.read_distance() < 15 and my_dog.ultrasonic.read_distance() >
↪ 1:
            my_dog.head_move([[0, 0, 0]], immediately=True, speed=90)
            my_dog.tail_move([[0]], immediately=True, speed=90)
            my_dog.rgb_strip.set_mode('bark', color='red', bps=2, brightness=0.8)
            my_dog.do_action('backward', step_count=1, speed=98)
            my_dog.wait_all_done()
            lean_forward()
            while len(my_dog.legs_action_buffer) > 0:
                sleep(0.1)
            my_dog.do_action('stand', step_count=1, speed=90)
            sleep(0.5)
        # relax
        if my_dog.dual_touch.read() != 'N':
            if len(my_dog.head_action_buffer) < 2:
                head_nod(1)
                my_dog.do_action('wag_tail', step_count=10, speed=80)
                my_dog.rgb_strip.set_mode('listen', color="#8A2BE2", bps=0.35,␣
↪brightness=0.8)
        # calm
        else:
            my_dog.rgb_strip.set_mode('breath', color='pink', bps=1, brightness=0.8)
            my_dog.tail_stop()
        sleep(0.2)

if __name__ == "__main__":
    try:
        alert()
    except KeyboardInterrupt:
        pass
    except Exception as e:
        print(f"\033[31mERROR: {e}\033[m")
    finally:
        my_dog.close()
```

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.

- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

2.3.5 5. Rest

PiDog will doze off on the ground, and when it hears sounds around it, it will stand up in confusion to see who woke it up.

Run the Code

```
cd ~/pidog/examples
sudo python3 5_rest.py
```

After the program runs, PiDog will get down on the ground, shake its head and tail as if dozing off. At the same time, its sound direction sensor module is working. If PiDog hears noise, it will stand up, look around, and then make a confused look. Then it'll doze off again.

Code

Note: You can **Modify/Reset/Copy/Run/Stop** the code below. But before that, you need to go to source code path like `pidog\examples`. After modifying the code, you can run it directly to see the effect.

```
#!/usr/bin/env python3
from pidog import Pidog
from time import sleep
from preset_actions import shake_head

my_dog = Pidog()
sleep(0.1)

def loop_around(amplitude=60, interval=0.5, speed=100):
    my_dog.head_move([[amplitude,0,0]], immediately=True, speed=speed)
    my_dog.wait_all_done()
    sleep(interval)
    my_dog.head_move([[-amplitude,0,0]], immediately=True, speed=speed)
    my_dog.wait_all_done()
    sleep(interval)
    my_dog.head_move([[0,0,0]], immediately=True, speed=speed)
    my_dog.wait_all_done()

def is_sound():
    if my_dog.ears.isdetected():
        direction = my_dog.ears.read()
        if direction != 0:
            return True
        else:
            return False
    else:
        return False
```

(continues on next page)

(continued from previous page)

```
def rest():
    my_dog.wait_all_done()
    my_dog.do_action('lie', speed=50)
    my_dog.wait_all_done()

    while True:
        # Sleeping
        my_dog.rgb_strip.set_mode('breath', 'pink', bps=0.3)
        my_dog.head_move([[0,0,-40]], immediately=True, speed=5)
        my_dog.do_action('doze_off', speed=92)
        # Cleanup sound detection
        sleep(1)
        is_sound()

        # keep sleeping
        while is_sound() is False:
            my_dog.do_action('doze_off', speed=92)
            sleep(0.2)

        # If heard anything, wake up
        # Set light to yellow and stand up
        my_dog.rgb_strip.set_mode('boom', 'yellow', bps=1)
        my_dog.body_stop()
        my_dog.do_action('stand', speed=90)
        my_dog.head_move([[0, 0, 0]], immediately=True, speed=80)
        my_dog.wait_all_done()
        # Look arround
        loop_around(60, 1, 60)
        sleep(0.5)
        # tilt head and being confused
        my_dog.speak('confused_3', volume=80)
        my_dog.do_action('tilting_head_left', speed=80)
        my_dog.wait_all_done()
        sleep(1.2)
        my_dog.head_move([[0, 0, -10]], immediately=True, speed=80)
        my_dog.wait_all_done()
        sleep(0.4)
        # Shake head , mean to ignore it
        shake_head(my_dog)
        sleep(0.2)

        # Lay down again
        my_dog.rgb_strip.set_mode('breath', 'pink', bps=1)
        my_dog.do_action('lie', speed=50)
        my_dog.wait_all_done()
        sleep(1)


if __name__ == "__main__":
    try:
        rest()
    except KeyboardInterrupt:
```

(continues on next page)

(continued from previous page)

```
        pass
    except Exception as e:
        print(f"\033[31mERROR: {e}\033[m")
    finally:
        my_dog.close()
```

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

2.3.6 6. Be Picked Up

Try lifting your PiDog from the ground, PiDog will feel like it can fly, and it will cheer in a superman pose.

Run the Code

```
cd ~/pidog/examples
sudo python3 6_be_picked_up.py
```

After the program runs, the 6-DOF IMU Module will always calculate the acceleration in the vertical direction. If PiDog is calculated to be in a state of weightlessness, PiDog assumes a superman pose and cheers. Otherwise, consider PiDog to be on flat ground and make a standing pose.

Code

Note: You can **Modify/Reset/Copy/Run/Stop** the code below. But before that, you need to go to source code path like `pidog\examples`. After modifying the code, you can run it directly to see the effect.

```
#!/usr/bin/env python3
from pidog import Pidog
from time import sleep

my_dog = Pidog()
sleep(0.1)


def fly():
    my_dog.rgb_strip.set_mode('boom', color='red', bps=3)
    my_dog.legs.servo_move([45, -45, 90, -80, 90, 90, -90, -90], speed=60)
```

(continues on next page)

(continued from previous page)

```
    my_dog.do_action('wag_tail', step_count=10, speed=100)
    my_dog.speak('woohoo', volume=80)
    my_dog.wait_legs_done()
    sleep(1)

def stand():
    my_dog.rgb_strip.set_mode('breath', color='green', bps=1)
    my_dog.do_action('stand', speed=60)
    my_dog.wait_legs_done()
    sleep(1)

def be_picked_up():
    isUp = False
    upflag = False
    downflag = False

    stand()

    while True:
        ax = my_dog.accData[0]
        print('ax: %s, is up: %s' % (ax, isUp))

        # gravity : 1G = -16384
        if ax < -18000: # if down, acceleration is in the same direction as gravity, ax
↪< -1G
            my_dog.body_stop()
            if upflag == False:
                upflag = True
            if downflag == True:
                isUp = False
                downflag = False
                stand()

        if ax > -13000: # if up, acceleration is the opposite of gravity, ax will > -1G
            my_dog.body_stop()
            if upflag == True:
                isUp = True
                upflag = False
                fly()
            if downflag == False:
                downflag = True

        sleep(0.02)


if __name__ == "__main__":
    try:
        be_picked_up()
    except KeyboardInterrupt:
        pass
    except Exception as e:
        print(f"\033[31mERROR: {e}\033[m")
```

(continues on next page)

(continued from previous page)

```
    finally:
        my_dog.close()
```

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

2.3.7 7. Face Track

PiDog will sit quietly in place. You applaud it, it looks your way, and if it sees you, it says hello.

Run the Code

```
cd ~/pidog/examples
sudo python3 7_face_track.py
```

After running this code, PiDog will start the camera and enable the face detection function. You can visit `http://` + `PiDog's IP` +`/mjpg` (like mine is `http://192.168.18.138:9000/mjpg`) in your browser to view the camera's picture.

Then PiDog will sit down and activate the Sound Direction Sensor Module to detect the direction of your clapping. When PiDog hears clapping (or other noise), it turns its head toward the sound source, trying to find you.

If it sees you (face detection finds an object), it will wag its tail and let out a bark.

Code

Note: You can **Modify/Reset/Copy/Run/Stop** the code below. But before that, you need to go to source code path like `pidog\examples`. After modifying the code, you can run it directly to see the effect.

```
#!/usr/bin/env python3
from pidog import Pidog
from time import sleep
from vilib import Vilib
from preset_actions import bark

my_dog = Pidog()
sleep(0.1)

def face_track():
```

(continues on next page)

(continued from previous page)

```
    Vilib.camera_start(vflip=False, hflip=False)
    Vilib.display(local=True, web=True)
    Vilib.human_detect_switch(True)
    sleep(0.2)
    print('start')
    yaw = 0
    roll = 0
    pitch = 0
    flag = False
    direction = 0

    my_dog.do_action('sit', speed=50)
    my_dog.head_move([[yaw, 0, pitch]], pitch_comp=-40, immediately=True, speed=80)
    my_dog.wait_all_done()
    sleep(0.5)
    # Cleanup sound detection by servos moving
    if my_dog.ears.isdetected():
        direction = my_dog.ears.read()

    while True:
        if flag == False:
            my_dog.rgb_strip.set_mode('breath', 'pink', bps=1)
        # If heard somthing, turn to face it
        if my_dog.ears.isdetected():
            flag = False
            direction = my_dog.ears.read()
            pitch = 0
            if direction > 0 and direction < 160:
                yaw = -direction
                if yaw < -80:
                    yaw = -80
            elif direction > 200 and direction < 360:
                yaw = 360 - direction
                if yaw > 80:
                    yaw = 80
            my_dog.head_move([[yaw, 0, pitch]], pitch_comp=-40, immediately=True,␣
↪speed=80)
            my_dog.wait_head_done()
            sleep(0.05)

        ex = Vilib.detect_obj_parameter['human_x'] - 320
        ey = Vilib.detect_obj_parameter['human_y'] - 240
        people = Vilib.detect_obj_parameter['human_n']

        # If see someone, bark at him/her
        if people > 0 and flag == False:
            flag = True
            my_dog.do_action('wag_tail', step_count=2, speed=100)
            bark(my_dog, [yaw, 0, 0], pitch_comp=-40, volume=80)
            if my_dog.ears.isdetected():
                direction = my_dog.ears.read()
```

(continues on next page)

(continued from previous page)

```
        if ex > 15 and yaw > -80:
            yaw -= 0.5 * int(ex/30.0+0.5)

        elif ex < -15 and yaw < 80:
            yaw += 0.5 * int(-ex/30.0+0.5)

        if ey > 25:
            pitch -= 1*int(ey/50+0.5)
            if pitch < - 30:
                pitch = -30
        elif ey < -25:
            pitch += 1*int(-ey/50+0.5)
            if pitch > 30:
                pitch = 30

        print('direction: %s |number: %s | ex, ey: %s, %s | yrp: %s, %s, %s '
            % (direction, people, ex, ey, round(yaw, 2), round(roll, 2), round(pitch,␣
↪2)),
            end='\r',
            flush=True,
            )
        my_dog.head_move([[yaw, 0, pitch]], pitch_comp=-40, immediately=True, speed=100)
        sleep(0.05)


if __name__ == "__main__":
    try:
        face_track()
    except KeyboardInterrupt:
        pass
    except Exception as e:
        print(f"\033[31mERROR: {e}\033[m")
    finally:
        Vilib.camera_close()
        my_dog.close()
```

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

2.3.8 8. Push Up

PiDog is an exercise-loving robot that will do push-ups with you.

Run the Code

```
cd ~/pidog/examples
sudo python3 8_pushup.py
```

After the program runs, PiDog will perform a plank, then cycle through push-ups and barks.

Code

Note: You can **Modify/Reset/Copy/Run/Stop** the code below. But before that, you need to go to source code path like `pidog\examples`. After modifying the code, you can run it directly to see the effect.

```
#!/usr/bin/env python3
from pidog import Pidog
from time import sleep
from preset_actions import push_up, bark

my_dog = Pidog()

sleep(0.5)


def main():
    my_dog.legs_move([[45, -25, -45, 25, 80, 70, -80, -70]], speed=50)
    my_dog.head_move([[0, 0, -20]], speed=90)
    my_dog.wait_all_done()
    sleep(0.5)
    bark(my_dog, [0, 0, -20])
    sleep(0.1)
    bark(my_dog, [0, 0, -20])

    sleep(1)
    my_dog.rgb_strip.set_mode("speak", color="blue", bps=2)
    while True:
        push_up(my_dog, speed=92)
        bark(my_dog, [0, 0, -40])
        sleep(0.4)


if __name__ == "__main__":
    try:
        main()
    except KeyboardInterrupt:
        pass
    except Exception as e:
        print(f"\033[31mERROR: {e}\033[m")
    finally:
        my_dog.close()
```

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

2.3.9 9. Howling

PiDog is not only a cute puppy, but also a mighty dog. Come hear it howl!

Run the Code

```
cd ~/pidog/examples
sudo python3 9_howling.py
```

After the program runs, PiDog will sit on the ground and howl.

Code

Note: You can **Modify/Reset/Copy/Run/Stop** the code below. But before that, you need to go to source code path like `pidog\examples`. After modifying the code, you can run it directly to see the effect.

```
#!/usr/bin/env python3
from pidog import Pidog
from time import sleep
from preset_actions import howling

my_dog = Pidog()

sleep(0.5)


def main():
    my_dog.do_action('sit', speed=50)
    my_dog.head_move([[0, 0, 0]], pitch_comp=-40, immediately=True, speed=80)
    sleep(0.5)
    while True:
        howling(my_dog)


if __name__ == "__main__":
    try:
```

(continues on next page)

(continued from previous page)

```
        main()
    except KeyboardInterrupt:
        pass
    except Exception as e:
        print(f"\033[31mERROR: {e}\033[m")
    finally:
        my_dog.close()
```

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

2.3.10 10. Balance

Because PiDog is equipped with a 6-DOF IMU module, it has a great sense of balance.

In this example, you can make PiDog walk smoothly on the table, even if you lift one side of the table, PiDog will walk smoothly on the gentle slope.

Run the Code

```
cd ~/pidog/examples
sudo python3 10_balance.py
```

After the program is running, you will see a printed keyboard on the terminal. You can control PiDog to walk smoothly on the ramp by typing the below keys.

Keys	Function
W	Forward
E	Stand
A	Turn Left
S	Backward
D	Turn Right
R	Each press slightly lifts the body; multiple presses are needed for a noticeable rise.
F	Each press lowers the body a bit; it takes multiple presses for a noticeable descent.

Code

Please find the code in .

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

2.3.11 11. Play PiDog with Keyboard

In this example, we will use the keyboard to control PiDog. You can press these keys in the terminal to make it act.

Keys	Function	Keys	Function	Keys	Function
1	doze off	q	bark harder	a	turn left
2	push-up	w	forward	s	backward
3	howling	e	pant	d	turn right
4	twist body	r	wag tail	f	shake head
5	scratch	t	hake head	g	high five
u	head roll	U	head roll+	z	lie
i	head pitch	I	head pitch+	x	stand up
o	head roll	O	head roll+	c	sit
j	head yaw	J	head yaw+	v	stretch
k	head pitch	K	head pitch+	m	head reset
l	head yaw	L	head yaw+	W	trot

Run the Code

```
cd ~/pidog/examples
sudo python3 11_keyboard_control.py
```

After the program runs, you will see a printed keyboard on the terminal. Now you can control PiDog with keyboard in terminal.

Code

Please find the code in .

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.

- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

2.3.12 12. Play PiDog with APP

In this lesson, you'll learn how to use the **SunFounder Controller App** to control your PiDog. This approach makes controlling your robotic dog more intuitive and interactive.

You need to download the APP on your phone/tablet first, then connect to the WLAN as PiDog, and finally create your own remote control on SunFounder Controller to control PiDog.

Control Pidog with App

To control PiDog via the SunFounder Controller App, follow these steps:

1. Install SunFounder Controller from **APP Store(iOS)** or **Google Play(Android)**.
2. Set Up Required Modules.
   - The `robot-hat`, `vilib`, and `pidog` modules need to be installed first, for details see: *Install All the Modules(Important)* section.
     - `robot-hat`
     - `vilib`
     - `pidog`
   - Then, install the `sunfounder-controller` module:

     ```
     cd ~
     git clone https://github.com/sunfounder/sunfounder-controller.git
     cd ~/sunfounder-controller
     sudo python3 setup.py install
     ```
3. Execute the following commands to start the control script:

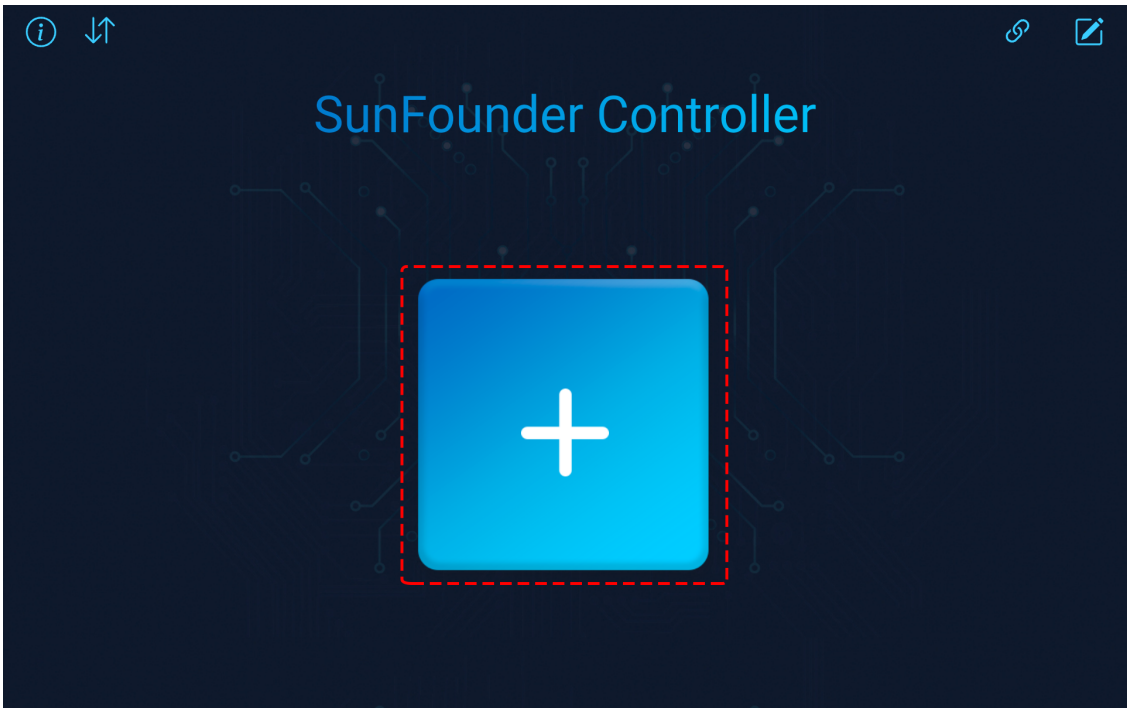
   ```
   cd ~/pidog/examples
   sudo python3 12_app_control.py
   ```

   Once the script runs successfully, you'll see a prompt like this:

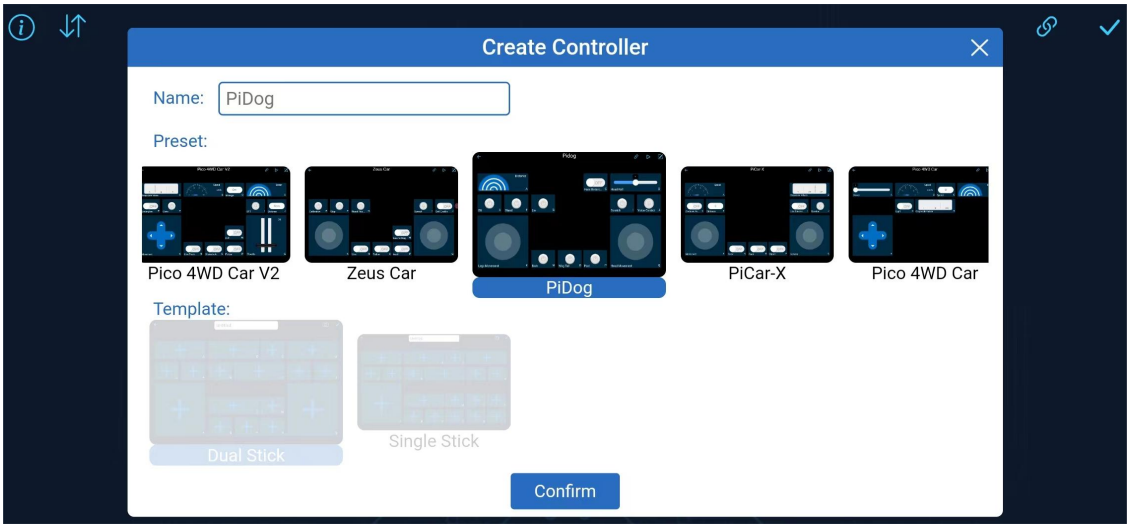
   ```
   Running on: http://192.168.18.138:9000/mjpg

   * Serving Flask app "vilib.vilib" (lazy loading)
   * Environment: development
   * Debug mode: off
   * Running on http://0.0.0.0:9000/ (Press CTRL+C to quit)
   ```

   This indicates that your PiDog is ready for network communication.
4. Connect PiDog and Sunfounder Controller.

- Connect your phone/tablet to the same WLAN as PiDog.
- Open the **Sunfounder Controller** APP. Click the + icon to add a controller.



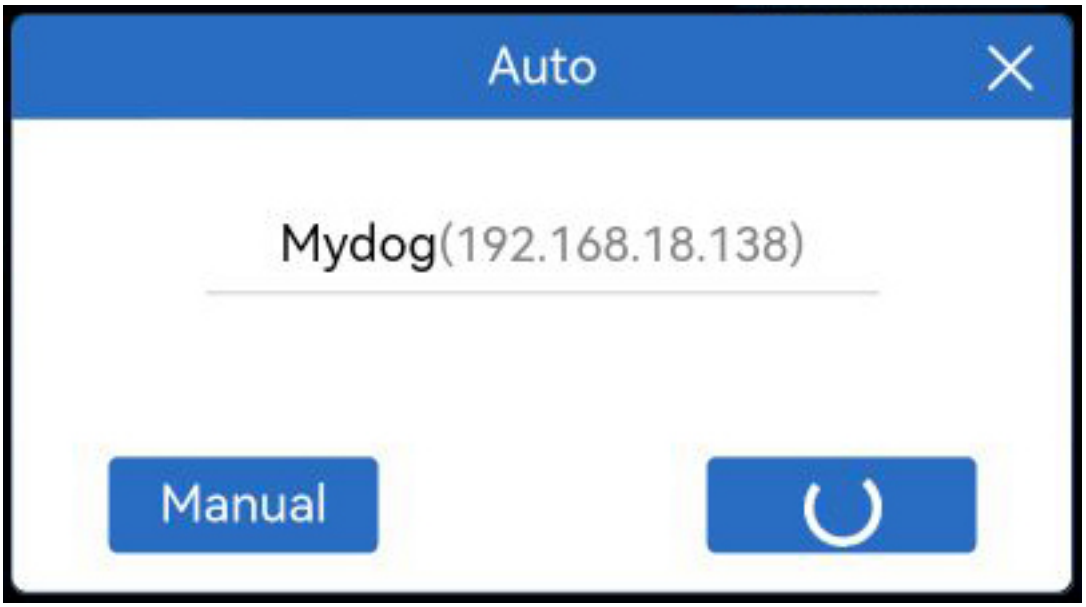
- Preset controllers are available for some products, here we choose **PiDog**. Give it a name, or simply tap **Confirm**.



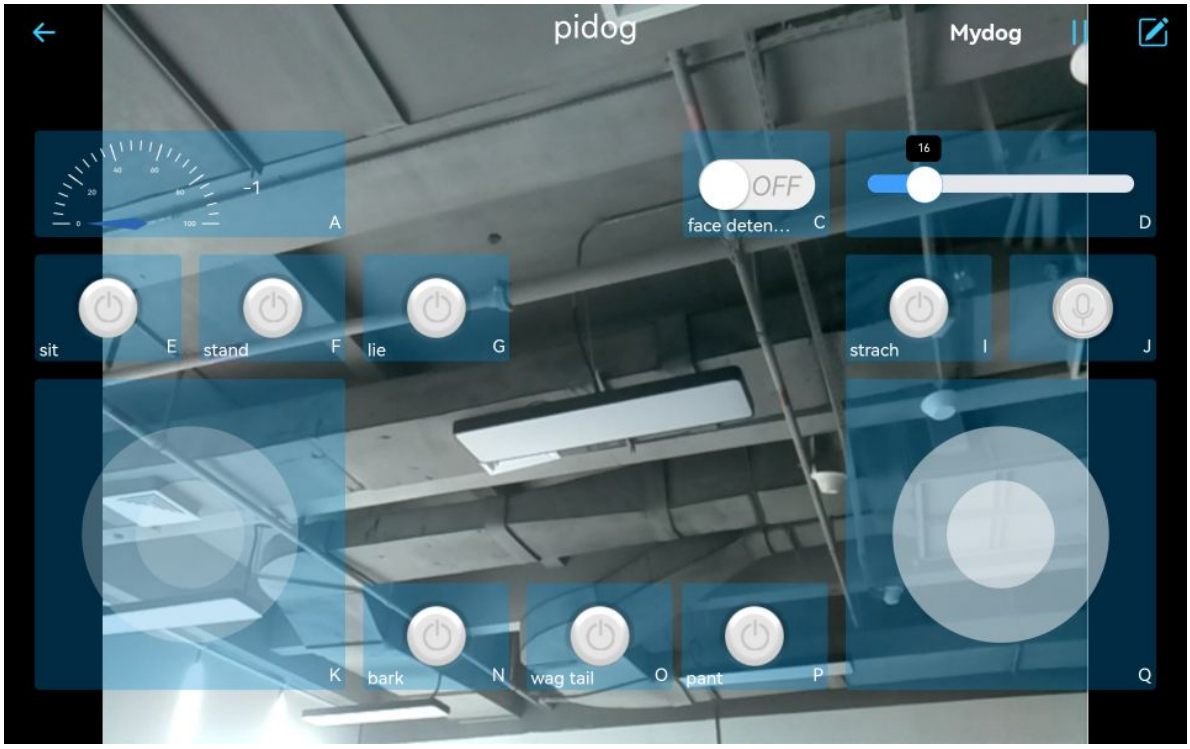
- Once inside, the app will automatically search for the **Mydog**. After a moment, you will see a prompt saying "Connected Successfully."

Note:

– You can also manually click the button. Wait a few seconds, MyDog(IP) will appear, click it to connect.



5. Run the Controller.
   - When the "Connected Successfully" prompt appears, tap the button in the upper right corner.
   - The picture taken by the camera will appear on the APP, and now you can control your PiDog with these widgets.



Here are the functions of the widgets.

- A: Detect the obstacle distance, that is, the reading of the ultrasonic module.
- C: Turn on/off face detection.
- D: Control PiDog's head tilt angle (tilt head).
- E: Sit.
- F: Stand.
- G: Lie.
- I: Scratch PiDog's head.
- N: Bark.
- O: Wag tail.
- P: Pant.
- K: Control PiDog's movement (forward, backward, left and right).
- Q: Controls the orientation of PiDog's head.
- J: Switch to voice control mode. It supports the following voice commands:
  - `forward`
  - `backward`
  - `turn left`
  - `turn right`
  - `trot`
  - `stop`

- lie down
- stand up
- sit
- bark
- bark harder
- pant
- wag tail
- shake head
- stretch
- doze off
- push-up
- howling
- twist body
- scratch
- handshake
- high five

Autostart PiDog on Boot

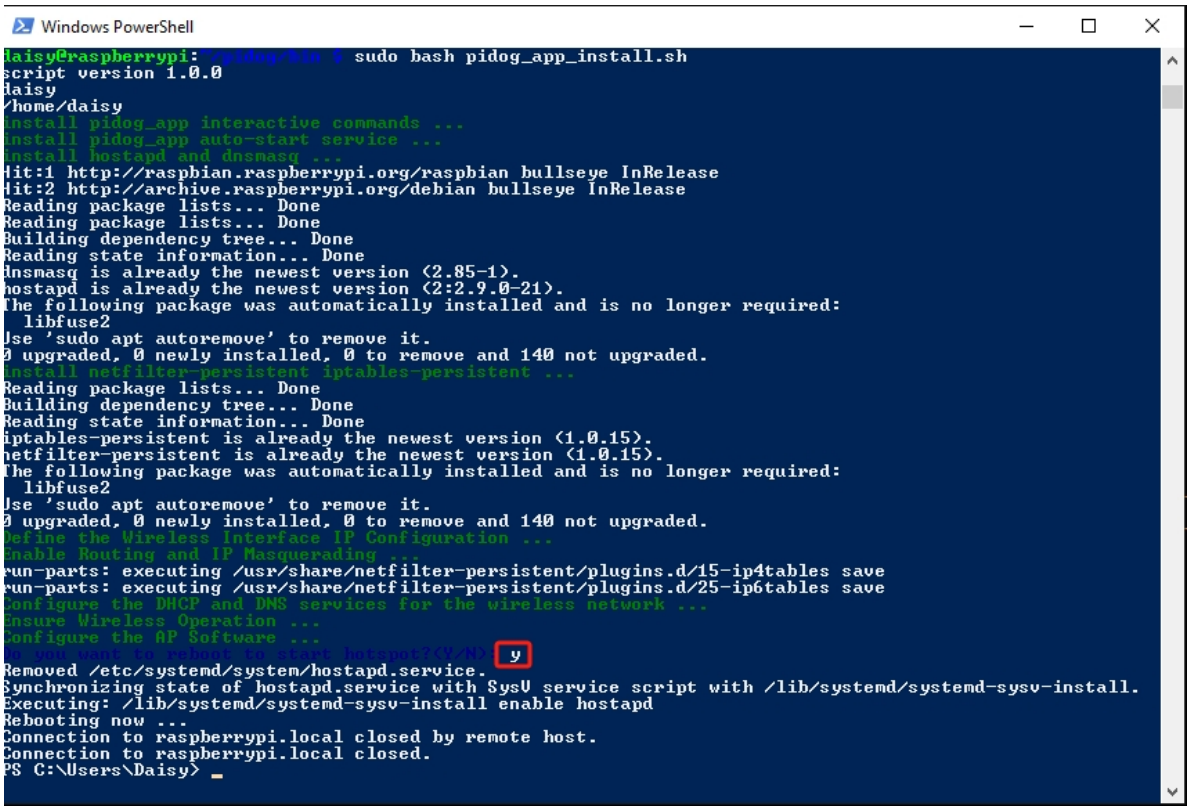
To avoid manually running the 12_app_control.py script every time, you can configure PiDog to start the script automatically upon boot:

How to set this up?

1. Execute the following commands to install and configure the `pidog_app` application:

```
cd ~/pidog/bin
sudo bash pidog_app_install.sh
```

2. When prompted, input `y` to reboot the PiDog.



3. After rebooting, PiDog will automatically start the control script. Then you can *Control Pidog with App*.

Warning: If you wish to run other scripts, first execute `pidog_app disable` to disable the autostart.

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

2.3.13 13. Ball Track

PiDog will sit quietly in place. You put a red ball in front of it, it will stand, and then chase the ball.

Run the Code

```
cd ~/pidog/examples
sudo python3 13_ball_track.py
```

After running this code, PiDog will start the camera. You can visit `http://+ PiDog's IP +/mjpg` (like mine is `http://192.168.18.138:9000/mjpg`) in your browser to view the camera's picture.

Code

Note: You can **Modify/Reset/Copy/Run/Stop** the code below. But before that, you need to go to source code path like `pidog\examples`. After modifying the code, you can run it directly to see the effect.

```
#!/usr/bin/env python3
from pidog import Pidog
from time import sleep
from vilib import Vilib
from preset_actions import bark

my_dog = Pidog()

sleep(0.1)

STEP = 0.5

def delay(time):
    my_dog.wait_legs_done()
    my_dog.wait_head_done()
    sleep(time)

def ball_track():
    Vilib.camera_start(vflip=False, hflip=False)
    Vilib.display(local=True, web=True)
    Vilib.color_detect_switch(True)
    sleep(0.2)
    print('start')
    yaw = 0
    roll = 0
    pitch = 0
    flag = False
    direction = 0

    my_dog.do_action('stand', speed=50)
    my_dog.head_move([[yaw, 0, pitch]], immediately=True, speed=80)
    delay(0.5)

    while True:

        ball_x = Vilib.detect_obj_parameter['color_x'] - 320
```

(continues on next page)

(continued from previous page)

```
        ball_y = Vilib.detect_obj_parameter['color_y'] - 240
        width = Vilib.detect_obj_parameter['color_w']

        if ball_x > 15 and yaw > -80:
            yaw -= STEP

        elif ball_x < -15 and yaw < 80:
            yaw += STEP

        if ball_y > 25:
            pitch -= STEP
            if pitch < - 40:
                pitch = -40
        elif ball_y < -25:
            pitch += STEP
            if pitch > 20:
                pitch = 20

        print(f"yaw: {yaw}, pitch: {pitch}, width: {width}")

        my_dog.head_move([[yaw, 0, pitch]], immediately=True, speed=100)
        if width == 0:
            pitch = 0
            yaw = 0
        elif width < 300:
            if my_dog.is_legs_done():
                if yaw < -30:
                    print("turn right")
                    my_dog.do_action('turn_right', speed=98)
                elif yaw > 30:
                    print("turn left")
                    my_dog.do_action('turn_left', speed=98)
                else:
                    my_dog.do_action('forward', speed=98)
        sleep(0.02)


if __name__ == "__main__":
    try:
        ball_track()
    except KeyboardInterrupt:
        pass
    except Exception as e:
        print(f"\033[31mERROR: {e}\033[m")
    finally:
        Vilib.camera_close()
        my_dog.close()
```

Then you may want to master its basic functions, or write some fun examples.

If you are familiar with Python programming, you can find examples of PiDog's basic functions in the `~/pidog/basic_examples` directory.

If you prefer, you can master them in a step-by-step fashion using the lessons provided below.

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

2.4 4. Easy Coding

Here, we delve into various functions, breaking them down for a comprehensive understanding. Each sub-topic is dedicated to a specific function, making it easier for you to grasp and implement them. Whether it's initiating parameters, controlling specific movements, or incorporating sensory inputs, we've covered them all. Navigate through the sub-topics below to kickstart your coding journey with Pidog.

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

2.4.1 1. PiDog Initialization

The functions of PiDog are written in the `Pidog` class, and the prototype of this class is shown below.

```
Class: Pidog()

__init__(leg_pins=DEFAULT_LEGS_PINS,
        head_pins=DEFAULT_HEAD_PINS,
        tail_pin=DEFAULT_TAIL_PIN,
        leg_init_angles=None,
        head_init_angles=None,
        tail_init_angle=None)
```

PiDog must be instantiated in one of several ways, as shown below.

1. Following are the simplest steps of initialization.

```
# Import Pidog class
from pidog import Pidog

# instantiate a Pidog
my_dog = Pidog()
```

2. PiDog has 12 servos, which can be initialized when we instantiate it.

```
# Import Pidog class
from pidog import Pidog

# instantiate a Pidog with custom initialized servo angles
my_dog = Pidog(leg_init_angles = [25, 25, -25, -25, 70, -45, -70, 45],
               head_init_angles = [0, 0, -25],
               tail_init_angle= [0]
           )
```

In the `Pidog` class, the servos are divided into three groups.

- `leg_init_angles` : In this array, 8 values determine the angles of eight servos, with the servos (pin numbers) they control being `2, 3, 7, 8, 0, 1, 10, 11`. From the foldout, you can see where these servos are located.
- `head_init_angles` : There is an array with 3 values, controllers for PiDog-head yaw, roll, pitch servos (`no. 4, 6, 5`) which react to yaw, roll, pitch, or Deflection of the body.
- `tail_init_angle` : In this array, there is only one value, which is dedicated to controlling the tail servo, which is 9.

3. `Pidog` allows you to redefine the serial number of the servos when instantiating the robot if your servo order is different.

```
# Import Pidog class
from pidog import Pidog

# instantiate a Pidog with custom initialized pins & servo angles
my_dog = Pidog(leg_pins=[2, 3, 7, 8, 0, 1, 10, 11],
               head_pins=[4, 6, 5],
               tail_pin=[9],
               leg_init_angles = [25, 25, -25, -25, 70, -45, -70, 45],
               head_init_angles = [0, 0, -25],
               tail_init_angle= [0]
           )
```

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

2.4.2 2. Leg Move

PiDog's leg movements are implemented by the following functions.

```
Pidog.legs_move(target_angles, immediately=True, speed=50)
```

- `target_angles`: It is a two-dimensional array composed of an array of 8 servo angles (referred to as angle group) as elements. These angle groups will be used to control the angles of the 8 foot servos. If multiple angle groups are written, the unexecuted angle groups will be stored in the cache.
- `immediately` : When calling the function, set this parameter to `True`, the cache will be cleared immediately to execute the newly written angle group; if the parameter is set to `False`, the newly written The incoming angular group is added to the execution queue.
- `speed` : The speed at which the angular group is executed.

Some common usages are listed below:

1. Take action immediately.

```
from pidog import Pidog
import time

my_dog = Pidog()

# half stand
my_dog.legs_move([[45, 10, -45, -10, 45, 10, -45, -10]], speed=50)
```

2. Add some angular groups to the execution queue.

```
from pidog import Pidog
import time

my_dog = Pidog()

# half stand
my_dog.legs_move([[45, 10, -45, -10, 45, 10, -45, -10]], speed=50)

# multiple actions
my_dog.legs_move([[45, 35, -45, -35, 80, 70, -80, -70],
                  [90, -30, -90, 30, 80, 70, -80, -70],
                  [45, 35, -45, -35, 80, 70, -80, -70]],  immediately=False, speed=30)
```

3. Perform repetitions within 10 seconds.

```
from pidog import Pidog
import time

my_dog = Pidog()

# half stand
my_dog.legs_move([[45, 10, -45, -10, 45, 10, -45, -10]], speed=50)
```

(continues on next page)

(continued from previous page)

```
# pushup preparation
my_dog.legs_move([[45, 35, -45, -35, 80, 70, -80, -70]], immediately=False, speed=20)

# pushup
for _ in range(99):
    my_dog.legs_move([[90, -30, -90, 30, 80, 70, -80, -70],
                      [45, 35, -45, -35, 80, 70, -80, -70]],  immediately=False,␣
↪speed=30)

# keep 10s
time.sleep(10)

# stop and half stand
my_dog.legs_move([[45, 10, -45, -10, 45, 10, -45, -10]], immediately=True, speed=50)
```

PiDog's leg control also has the following functions that can be used together:

```
Pidog.is_legs_done()
```

This function is used to determine whether the angle group in the cache has been executed. If yes, return `True`; otherwise, return `False`.

```
Pidog.wait_legs_done()
```

Suspends the program until the angle groups in the cache have been executed.

```
Pidog.legs_stop()
```

Empty the angular group in the cache.

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

2.4.3 3. Head Move

The control of PiDog's head servo is implemented by the following functions.

```
Pidog.head_move(target_yrps, roll_comp=0, pitch_comp=0, immediately=True, speed=50)
```

- `target_angles` : It is a two-dimensional array composed of an array of 3 servo angles (referred to as angle group) as elements. These angle groups will be used to control the angles of the 8 foot servos. If multiple angle groups are written, the unexecuted angle groups will be stored in the cache.
- `roll_comp` : Provides angular compensation on the roll axis.
- `pitch_comp` : Provides angle compensation on the pitch axis.
- `immediately` : When calling the function, set this parameter to `True`, the cache will be cleared immediately to execute the newly written angle group; if the parameter is set to `False`, the newly written The incoming angular group is added to the execution queue.
- `speed` : The speed at which the angular group is executed.

PiDog's head servo control also has some supporting functions:

```
Pidog.is_head_done()
```

Whether all the head actions in the buffer to be executed

```
Pidog.wait_head_done()
```

Wait for all the head actions in the buffer to be executed

```
Pidog.head_stop()
```

Clear all the head actions of leg in the buffer, to make head servos stop

Here are some common use cases:

1. Nod five times.

```
from pidog import Pidog
import time

my_dog = Pidog()

for _ in range(5):
    my_dog.head_move([[0, 0, 30],[0, 0, -30]], speed=80)
    my_dog.wait_head_done()
    time.sleep(0.5)
```

2. Shake your head for 10 seconds.

```
from pidog import Pidog
import time

my_dog = Pidog()

for _ in range(99):
    my_dog.head_move([[30, 0, 0],[-30, 0, 0]], immediately=False, speed=30)
```

(continues on next page)

(continued from previous page)

```
# keep 10s
time.sleep(10)

my_dog.head_move([[0, 0, 0]], immediately=True, speed=80)
```

3. Whether sitting or half standing, PiDog keeps its head level when shaking its head.

```
from pidog import Pidog
import time

my_dog = Pidog()

# action list
shake_head = [[30, 0, 0],[-30, 0, 0]]
half_stand_leg = [[45, 10, -45, -10, 45, 10, -45, -10]]
sit_leg = [[30, 60, -30, -60, 80, -45, -80, 45]]

while True:
    # shake head in half stand
    my_dog.legs_move(half_stand_leg, speed=30)
    for _ in range(5):
        my_dog.head_move(shake_head, pitch_comp=0, speed=50)
    my_dog.wait_head_done()
    time.sleep(0.5)

    # shake head in sit
    my_dog.legs_move(sit_leg, speed=30)
    for _ in range(5):
        my_dog.head_move(shake_head, pitch_comp=-30, speed=50)
    my_dog.wait_head_done()
    time.sleep(0.5)
```

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

2.4.4 4. Tail Move

Following are the functions that control PiDog's tail. This function is similar to *2. Leg Move*.

```
Pidog.tail_move(target_angles, immediately=True, speed=50)
```

- `target_angles` : It is a two-dimensional array composed of an array of 1 servo angles (referred to as angle group) as elements. These angle groups will be used to control the angles of the 8 foot servos. If multiple angle groups are written, the unexecuted angle groups will be stored in the cache.
- `immediately` : When calling the function, set this parameter to `True`, the cache will be cleared immediately to execute the newly written angle group; if the parameter is set to `False`, the newly written The incoming angular group is added to the execution queue.
- `speed` : The speed at which the angular group is executed.

PiDog's tail servo control also has some supporting functions:

```
Pidog.is_tail_done()
```

whether all the tail actions in the buffer to be executed

```
Pidog.wait_tail_done()
```

wait for all the tail actions in the buffer to be executed

```
Pidog.tail_stop()
```

clear all the tail actions of leg in the buffer, to make tail servo stop

Here are some common usages:

1. Wag tail for 10 seconds.

```
from pidog import Pidog
import time

my_dog = Pidog()

for _ in range(99):
    my_dog.tail_move([[30],[-30]], immediately=False, speed=30)

# keep 10s
time.sleep(10)

my_dog.tail_stop()
```

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.

- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

2.4.5 5. Stop All Actions

After the previous chapters, you can find that the servo control of PiDog is divided into three threads. This allows PiDog's head and body to move at the same time, even with two lines of code.

Here are a few functions that work with the three servo threads:

```
Pidog.wait_all_done()
```

Wait for all the actions in the leg actions buffer, head buffer and tail buffer to be executed

```
Pidog.body_stop()
```

Stop all the actions of legs, head and tail

```
Pidog.stop_and_lie()
```

Stop all the actions of legs, head and tail, then reset to "lie" pose

```
Pidog.close()
```

Stop all the actions, reset to "lie" pose, and close all the threads, usually used when exiting a program

Here are some common usages:

```
from pidog import Pidog
import time

my_dog = Pidog()

try:
    # pushup prepare
    my_dog.legs_move([[45, 35, -45, -35, 80, 70, -80, -70]], speed=30)
    my_dog.head_move([[0, 0, 0]], pitch_comp=-10, speed=80)
    my_dog.wait_all_done() # wait all the actions to be done
    time.sleep(0.5)

    # pushup
    leg_pushup_action = [
        [90, -30, -90, 30, 80, 70, -80, -70],
        [45, 35, -45, -35, 80, 70, -80, -70],
    ]
    head_pushup_action = [
        [0, 0, -30],
        [0, 0, 20],
    ]

    # fill action buffers
    for _ in range(50):
        my_dog.legs_move(leg_pushup_action, immediately=False, speed=50)
```

(continues on next page)

(continued from previous page)

```
        my_dog.head_move(head_pushup_action, pitch_comp=-10, immediately=False, speed=50)

    # show buffer length
    print(f"legs buffer length (start): {len(my_dog.legs_action_buffer)}")

    # keep 5 second & show buffer length
    time.sleep(5)
    print(f"legs buffer length (5s): {len(my_dog.legs_action_buffer)}")

    # stop action & show buffer length
    my_dog.stop_and_lie()
    print(f"legs buffer length (stop): {len(my_dog.legs_action_buffer)}")

except KeyboardInterrupt:
    pass
except Exception as e:
    print(f"\033[31mERROR: {e}\033[m")
finally:
    print("closing ...")
    my_dog.close() # close all the servo threads
```

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

2.4.6 6. Do Preset Action

Some commonly used actions have been pre-written in PiDog's library. You can call the following function to make PiDog do these actions directly.

```
Pidog.do_action(action_name, step_count=1, speed=50)
```

- `action_name` : Action name, the following strings can be written.
  - "sit"
  - "half_sit"
  - "stand"
  - "lie"
  - "lie_with_hands_out"

  - "forward"
  - "backward"
  - "turn_left"
  - "turn_right"
  - "trot"
  - "stretch"
  - "push_up"
  - "doze_off"
  - "nod_lethargy"
  - "shake_head"
  - "tilting_head_left"
  - "tilting_head_right"
  - "tilting_head"
  - "head_bark"
  - "head_up_down"
  - "wag_tail"

- `step_count` : How many times to perform this action.
- `speed` : How fast to perform the action.

Here is an example of usage:

1. Do ten push-ups, then sit on the floor and act cute.

```
from pidog import Pidog
import time

my_dog = Pidog()

try:
    # pushup
    my_dog.do_action("half_sit", speed=60)
    my_dog.do_action("push_up", step_count=10, speed=60)
    my_dog.wait_all_done()

    # act cute
    my_dog.do_action("sit", speed=60)
    my_dog.do_action("wag_tail", step_count=100,speed=90)
    my_dog.do_action("tilting_head", step_count=5, speed=20)
    my_dog.wait_head_done()

    my_dog.stop_and_lie()

except KeyboardInterrupt:
    pass
except Exception as e:
    print(f"\033[31mERROR: {e}\033[m")
```

(continues on next page)

(continued from previous page)

```
finally:
    print("closing ...")
    my_dog.close()
```

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

2.4.7 7. PiDog Speak

PiDog can make sound, it is actually playing a piece of audio.

These audios are saved under `pidog\sounds` path, you can call the following function to play them.

```
Pidog.speak(name)
```

- `name` : Filename (without suffix), such as `"angry"`. `Pidog` provides the following audio.
  - `"angry"`
  - `"confused_1"`
  - `"confused_2"`
  - `"confused_3"`
  - `"growl_1"`
  - `"growl_2"`
  - `"howling"`
  - `"pant"`
  - `"single_bark_1"`
  - `"single_bark_2"`
  - `"snoring"`
  - `"woohoo"`

Here is an example of usage:

```
# !/usr/bin/env python3
''' play sound effecfs
    Note that you need to run with "sudo"
API:
    Pidog.speak(name, volume=100)
        play sound effecf in the file "../sounds"
        - name    str, file name of sound effect, no suffix required, eg: "angry"
        - volume  int, volume 0-100, default 100
'''
from pidog import Pidog
import os
import time

# change working directory
abspath = os.path.abspath(os.path.dirname(__file__))
# print(abspath)
os.chdir(abspath)

my_dog = Pidog()

print("\033[033mNote that you need to run with \"sudo\", otherwise there may be no sound.
↪\033[m")

# my_dog.speak("angry")
# time.sleep(2)

for name in os.listdir('../sounds'):
    name = name.split('.')[0] # remove suffix
    print(name)
    my_dog.speak(name)
    # my_dog.speak(name, volume=50)
    time.sleep(3) # Note that the duration of each sound effect is different
print("closing ...")
my_dog.close()
```

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

2.4.8 8. Read Distance

Through the Ultrasonic Module in its head, PiDog can detect obstacles ahead.

An ultrasonic module can detect objects between 2 and 400 cm away.

With the following function, you can read the distance as a floating point number.

```
Pidog.ultrasonic.read_distance()
```

Here is an example of usage:

```
from pidog import Pidog
import time

my_dog = Pidog()
while True:
    distance = my_dog.ultrasonic.read_distance()
    distance = round(distance,2)
    print(f"Distance: {distance} cm")
    time.sleep(0.5)
```

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

2.4.9 9. PiDog RGB Strip

There is an RGB Strip on PiDog's chest, which PiDog can use to express emotions.

You can call the following function to control it.

```
Pidog.rgb_strip.set_mode(style='breath', color='white', bps=1, brightness=1):
```

- `style` : The lighting display mode of RGB Strip, the following are its available values.
  - `breath`
  - `boom`
  - `bark`
- `color` : The lights of the RGB Strip show the colors. You can enter 16-bit RGB values, such as `#a10a0a`, or the following color names.
  - `"white"`

  - "black"
  - "white"
  - "red"
  - "yellow"
  - "green"
  - "blue"
  - "cyan"
  - "magenta"
  - "pink"

- `brightness` : RGB Strip lights display brightness, you can enter a floating-point value from 0 to 1, such as `0.5`.
- `delay` : Float, display animation speed, the smaller the value, the faster the change.

Use the following statement to disable RGB Striping.

```
Pidog.rgb_strip.close()
```

Here are examples of their use:

```
from pidog import Pidog
import time

my_dog = Pidog()

while True:
    # style="breath", color="pink"
    my_dog.rgb_strip.set_mode(style="breath", color='pink')
    time.sleep(3)

    # style:"boom", color="#a10a0a"
    my_dog.rgb_strip.set_mode(style="bark", color="#a10a0a")
    time.sleep(3)

    # style:"boom", color="#a10a0a", brightness=0.5, bps=2.5
    my_dog.rgb_strip.set_mode(style="boom", color="#a10a0a", bps=2.5, brightness=0.5)
    time.sleep(3)

    # close
    my_dog.rgb_strip.close()
    time.sleep(2)
```

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.

- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

2.4.10 10. IMU Read

Through the 6-DOF IMU Module, PiDog can determine if it's standing on a slope, or if it's being picked up.

The 6-DOF IMU Module is equipped with a 3-axis accelerometer and a 3-axis gyroscope, allowing acceleration and angular velocity to be measured in three directions.

Note: Before using the module, make sure that it is correctly assembled. The label on the module will let you know if it is reversed.

You can read their acceleration with:

```
ax, ay, az = Pidog.accData
```

With the PiDog placed horizontally, the acceleration on the x-axis (ie ax) should be close to the acceleration of gravity (1g), with a value of -16384. The values of the y-axis and x-axis are close to 0.

Use the following way to read their angular velocity:

```
gx, gy, gz = my_dog.gyroData
```

In the case where PiDog is placed horizontally, all three values are close to 0.

Here are some examples of how 6-DOF Module is used:

1. Read real-time acceleration, angular velocity

```
from pidog import Pidog
import time

my_dog = Pidog()

my_dog.do_action("pushup", step_count=10, speed=20)

while True:
    ax, ay, az = my_dog.accData
    gx, gy, gz = my_dog.gyroData
    print(f"accData: {ax/16384:.2f} g ,{ay/16384:.2f} g, {az/16384:.2f} g      ␣
↪gyroData: {gx} °/s, {gy} °/s, {gz} °/s")
    time.sleep(0.2)
    if my_dog.is_legs_done():
        break

my_dog.stop_and_lie()

my_dog.close()
```

2. Calculate the lean angle of PiDog's body.

```
from pidog import Pidog
import time
import math

my_dog = Pidog()

while True:
    ax, ay, az = my_dog.accData
    body_pitch = math.atan2(ay,ax)/math.pi*180%360-180
    print(f"Body Degree: {body_pitch:.2f} °" )
    time.sleep(0.2)

my_dog.close()
```

3. While leaning, PiDog keeps its eyes level.

```
from pidog import Pidog
import time
import math

my_dog = Pidog()

while True:
    ax, ay, az = my_dog.accData
    body_pitch = math.atan2(ay,ax)/math.pi*180%360-180
    my_dog.head_move([[0, 0, 0]], pitch_comp=-body_pitch, speed=80)
    time.sleep(0.2)

my_dog.close()
```

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

2.4.11 11. Sound Direction Detect

The PiDog has a Sound Direction Sensor Module that detects where sound is coming from, and we can trigger it by clapping near it.

Using this module is as simple as calling these functions.

```
Pidog.ears.isdetected()
```

Returns `True` if sound is detected, `False` otherwise.

```
Pidog.ears.read()
```

This function returns the direction of the sound source, with a range of 0 to 359; if the sound comes from the front, it returns 0; if it comes from the right, it returns 90.

An example of how to use this module is as follows:

```
from pidog import Pidog

my_dog = Pidog()

while True:
    if my_dog.ears.isdetected():
        direction = my_dog.ears.read()
        print(f"sound direction: {direction}")
```

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

2.4.12 12. Pat the PiDog's Head

The Touch Swich on the head of PiDog can detect how you touch it. You can call the following functions to use it.

```
Pidog.dual_touch.read()
```

- Touch the module from left to right (front to back for PiDog's orientation), it will return `"LS"`.
- Touch the module from right to left, it will return `"RS"`.
- Touch the module If the left side of the module is touched, it will return `"L"`.
- If the right side of the module is touched, it will return `"R"`.

- If the module is not touched, it will return "N".

Here is an example of its use:

```
from pidog import Pidog
import time

my_dog = Pidog()
while True:
    touch_status = my_dog.dual_touch.read()
    print(f"touch_status: {touch_status}")
    time.sleep(0.5)
```

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

2.4.13 13. More

The following address will explain the use of PiDog's more basic functions:

- Vilib Library

  Vilib is a library developed by SunFounder for Raspberry Pi camera.

  It contains some practical functions, such as taking pictures, video recording, pose detection, face detection, motion detection, image classification and so on.

- SunFounder Controller

  SunFounder Controller is an application that allows users to customize the controller for controlling their robot or as an IoT platform.

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

CHAPTER

THREE

SEE · HEAR · RESPOND— AI-POWERED WITH MULTI-LLMS

Go beyond movement and vision by adding **speech** and **AI**. Here you will explore text-to-speech (TTS), speech-to-text (STT), and large language models (LLMs) to make your Fusion HAT talk, listen, and even chat with you like a smart robot.

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

3.1 14. TTS with Espeak and Pico2Wave

In this lesson, we'll use two built-in text-to-speech (TTS) engines on Raspberry Pi — **Espeak** and **Pico2Wave** — to make the Pidog talk.

These two engines are both simple and run offline, but they sound quite different:

- **Espeak**: very lightweight and fast, but the voice is robotic. You can adjust speed, pitch, and volume.
- **Pico2Wave**: produces a smoother and more natural voice than Espeak, but has fewer options to configure.

You'll hear the difference in **voice quality** and **features**.

3.1.1 Before You Start

Make sure you've completed:

- *Install All the Modules(Important)* — Install `robot-hat`, `vilib`, `pidog` modules, then run the script `i2samp.sh`.

3.1.2 Testing Espeak

Espeak is a lightweight TTS engine included in Raspberry Pi OS. Its voice sounds robotic, but it is highly configurable: you can adjust volume, pitch, speed, and more.

Steps to try it out:

- Create a new file with the command:

```
cd ~/pidog/examples
sudo nano test_tts_espeak.py
```

- Then copy the example code into it. Press `Ctrl+X`, then `Y`, and finally `Enter` to save and exit.

```
from pidog.tts import Espeak

tts = Espeak()

# Optional voice tuning
# tts.set_amp(100)   # 0 to 200
# tts.set_speed(150) # 80 to 260
# tts.set_gap(5)     # 0 to 200
# tts.set_pitch(50)  # 0 to 99

# Quick hello (sanity check)
tts.say("Hello! I'm Espeak TTS.")
```

- Run the program with:

```
sudo python3 test_tts_espeak.py
```

- You should hear the Pidog say: "Hello! I'm Espeak TTS."
- Uncomment the voice tuning lines in the code to experiment with how `amp`, `speed`, `gap`, and `pitch` affect the sound.

3.1.3 Testing Pico2Wave

Pico2Wave produces a more natural, human-like voice than Espeak. It's simpler to use but less flexible — you can only change the language, not the pitch or speed.

Steps to try it out:

- Create a new file with the command:

```
cd ~/pidog/examples
sudo nano test_tts_pico2wave.py
```

- Then copy the example code into it. Press `Ctrl+X`, then `Y`, and finally `Enter` to save and exit.

```
from pidog.tts import Pico2Wave

tts = Pico2Wave()

tts.set_lang('en-US')  # en-US, en-GB, de-DE, es-ES, fr-FR, it-IT

# Quick hello (sanity check)
tts.say("Hello! I'm Pico2Wave TTS.")
```

- Run the program with:

```
sudo python3 test_tts_pico2wave.py
```

- You should hear the Pidog say: "Hello! I'm Pico2Wave TTS."
- Try switching the language (for example, `es-ES` for Spanish) and listen to the difference.

3.1.4 Troubleshooting

- **No sound when running Espeak or Pico2Wave**
  - Check that your speakers/headphones are connected and volume is not muted.
  - Run a quick test in terminal:

    ```
    espeak "Hello world"
    pico2wave -w test.wav "Hello world" && aplay test.wav
    ```

  If you hear nothing, the issue is with audio output, not your Python code.
- **Espeak voice sounds too fast or too robotic**
  - Try adjusting the parameters in your code:

    ```
    tts.set_speed(120)   # slower
    tts.set_pitch(60)    # different pitch
    ```

- **Permission denied when running code**
  - Try running with `sudo`:

    ```
    sudo python3 test_tts_espeak.py
    ```

3.1.5 Comparison: Espeak vs Pico2Wave

Feature	Espeak	Pico2Wave
Voice quality	Robotic, synthetic	More natural, human-like
Languages	Default English	Fewer, but common ones
Adjustable	Yes (speed, pitch, etc.)	No (only language)
Performance	Very fast, lightweight	Slightly slower, heavier

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

3.2 15. TTS with Piper and OpenAI

In the previous lesson, we tried two built-in TTS engines on Raspberry Pi (**Espeak** and **Pico2Wave**). Now let's explore two more powerful options: **Piper** (offline, neural network-based) and **OpenAI TTS** (online, cloud-based).

- **Piper**: a local TTS engine that runs offline on Raspberry Pi.
- **OpenAI TTS**: an online service that provides very natural, human-like voices.

3.2.1 Before You Start

Make sure you've completed:

- *Install All the Modules(Important)* — Install `robot-hat`, `vilib`, `pidog` modules, then run the script `i2samp.sh`.

3.2.2 Testing Piper

Steps to try it out:

1. Create a new file:

```
cd ~/pidog/examples
sudo nano test_tts_piper.py
```

2. Copy the example code below into the file. Press `Ctrl+X`, then `Y`, and finally `Enter` to save and exit.

```
from pidog.tts import Piper

tts = Piper()

# List supported languages
print(tts.available_countrys())

# List models for English (en_us)
print(tts.available_models('en_us'))

```

(continues on next page)

(continued from previous page)

```
# Set a voice model (auto-download if not already present)
tts.set_model("en_US-amy-low")

# Say something
tts.say("Hello! I'm Piper TTS.")
```

- `available_countrys()`: print supported languages.
- `available_models()`: list available models for that language.
- `set_model()`: set the voice model (downloads automatically if missing).
- `say()`: convert text to speech and play it.

3. Run the program:

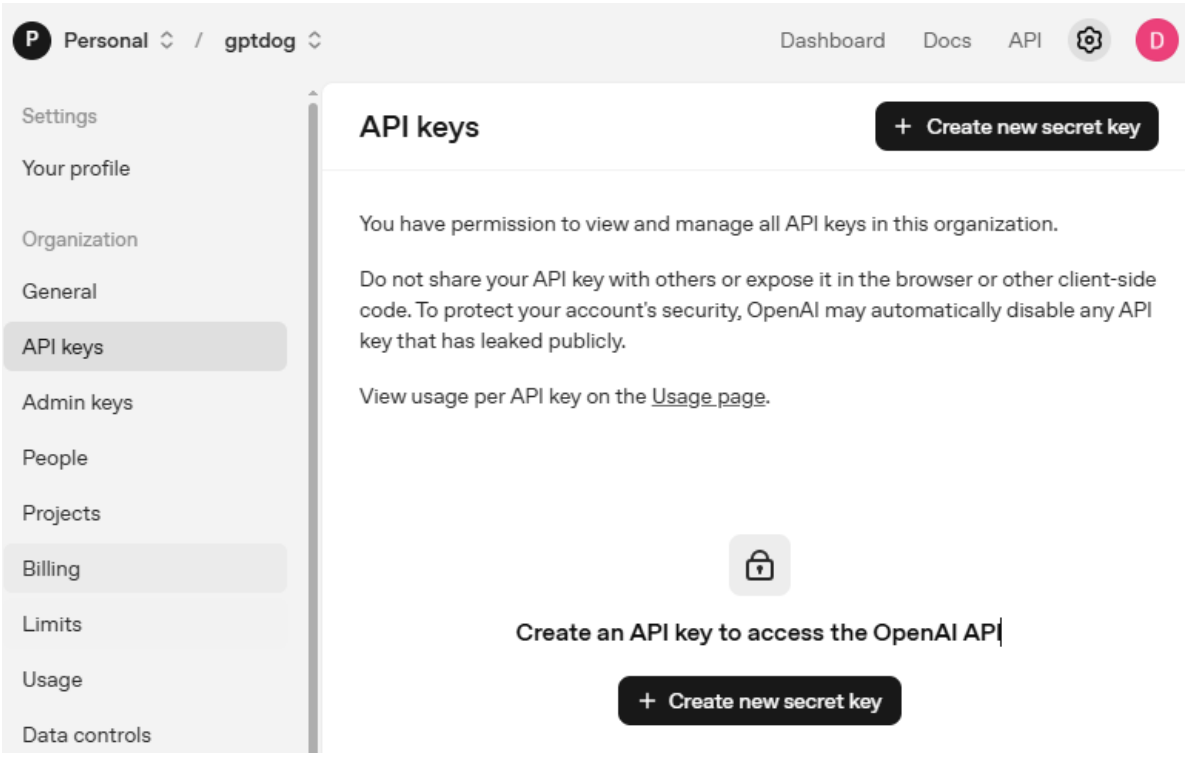
```
sudo python3 test_tts_piper.py
```

4. The first time you run it, the selected voice model will be downloaded automatically.
   - You should then hear the Pidog say: `Hello! I'm Piper TTS.`
   - You can change to another language model by calling `set_model()` with a different name.

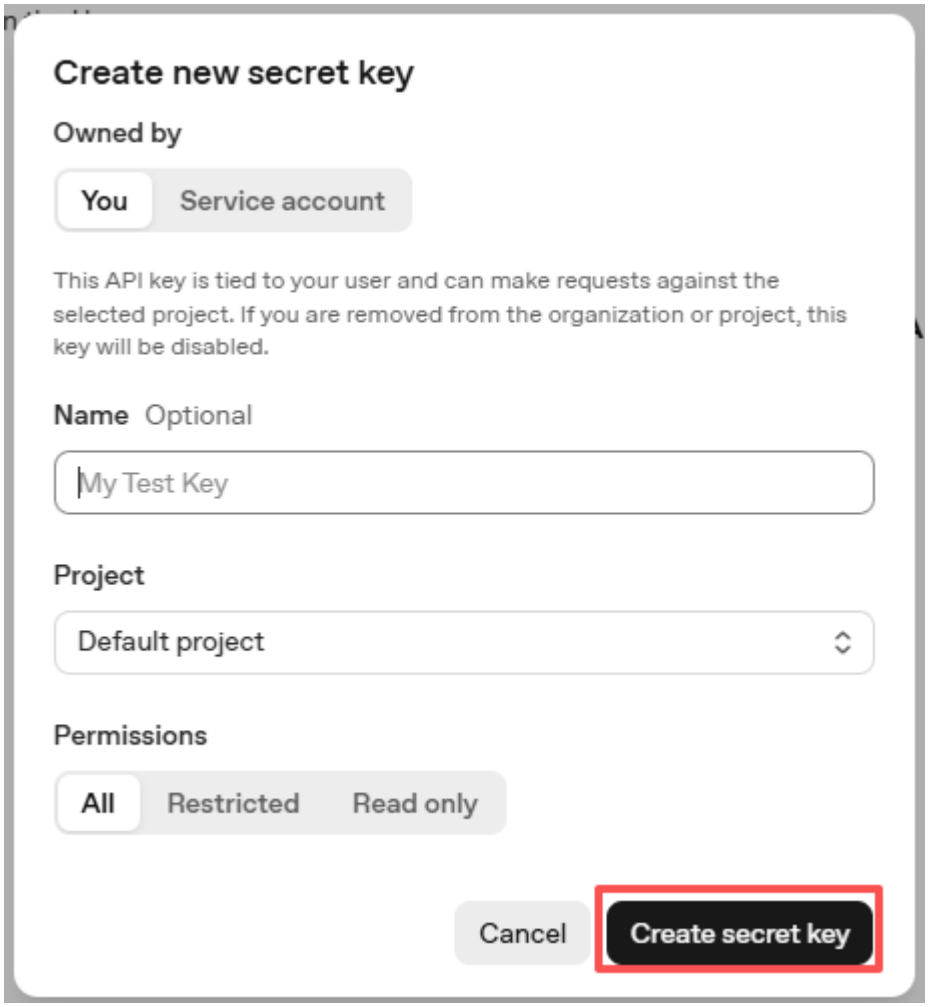
3.2.3 Testing OpenAI TTS

Get and save your API Key

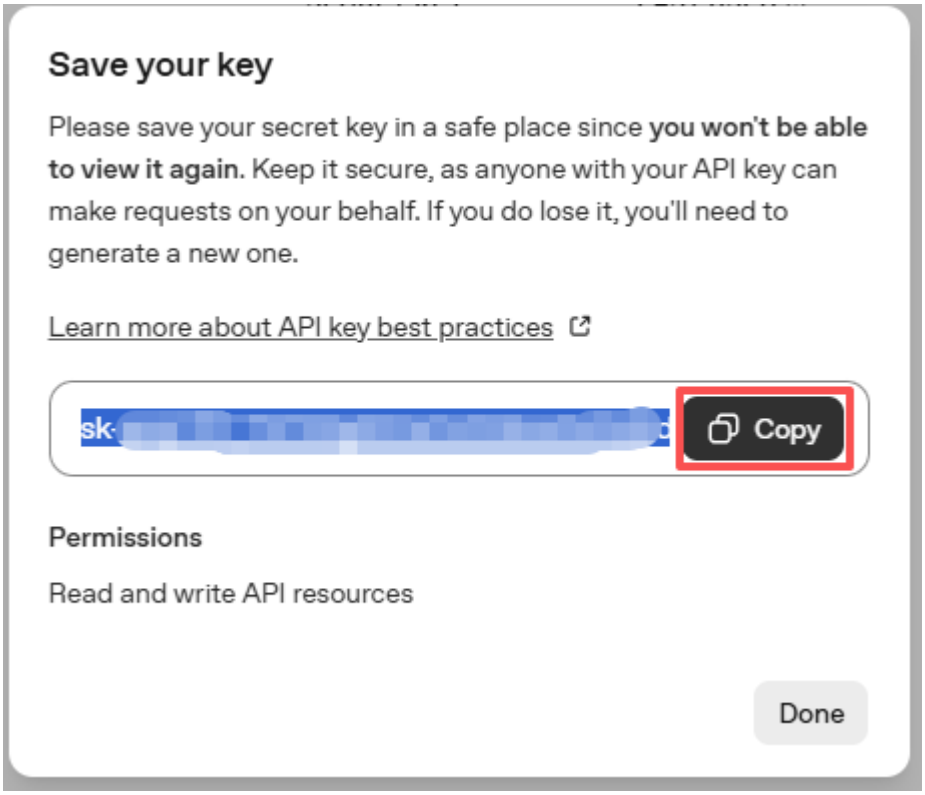
1. Go to and log in. On the **API keys** page, click **Create new secret key**.



2. Fill in the details (Owner, Name, Project, and permissions if needed), then click **Create secret key**.



3. Once the key is created, copy it right away — you won't be able to see it again. If you lose it, you must generate a new one.



4. In your project folder (for example: `/pidog/examples`), create a file called `secret.py`:

```
cd ~/pidog/examples
sudo nano secret.py
```

5. Paste your key into the file like this:

```
# secret.py
# Store secrets here. Never commit this file to Git.
OPENAI_API_KEY = "sk-xxx"
```

Write and Run a Test Program

1. Create a new file:

```
cd ~/pidog/examples
sudo nano test_tts_openai.py
```

2. Copy the example code below into the file. Press `Ctrl+X`, then `Y`, and finally `Enter` to save and exit.

```
from pidog.tts import OpenAI_TTS
from secret import OPENAI_API_KEY   # or use the try/except version shown above

# Initialize OpenAI TTS
tts = OpenAI_TTS(api_key=OPENAI_API_KEY)
tts.set_model('gpt-4o-mini-tts')  # low-latency TTS model
tts.set_voice('alloy')            # pick a voice

# Quick hello (sanity check)
tts.say("Hello! I'm OpenAI TTS.")
```

3. Run the program:

```
sudo python3 test_tts_openai.py
```

4. You should hear the Pidog say:

   `Hello! I'm OpenAI TTS.`

3.2.4 Troubleshooting

- **No module named 'secret'**

  This means `secret.py` is not in the same folder as your Python file. Move `secret.py` into the same directory where you run the script, e.g.:

```
ls ~/pidog/examples
# Make sure you see both: secret.py and your .py file
```

- **OpenAI: Invalid API key / 401**
  - Check that you pasted the full key (starts with `sk-`) and there are no extra spaces/newlines.
  - Ensure your code imports it correctly:

```
from secret import OPENAI_API_KEY
```

  - Confirm network access on your Pi (try `ping api.openai.com`).
- **OpenAI: Quota exceeded / billing error**
  - You may need to add billing or increase quota in the OpenAI dashboard.
  - Try again after resolving the account/billing issue.
- **Piper: tts.say() runs but no sound**
  - Make sure a voice model is actually present:

```
ls ~/.local/share/piper/voices
```

  - Confirm your model name matches exactly in code:

```
tts.set_model("en_US-amy-low")
```

  - Check the audio output device/volume on your Pi (`alsamixer`), and that speakers are connected and powered.
- **ALSA / sound device errors (e.g., "Audio device busy" or "No such file or directory")**
  - Close other programs using audio.
  - Reboot the Pi if the device stays busy.
  - For HDMI vs. headphone jack output, select the correct device in Raspberry Pi OS audio settings.
- **Permission denied when running Python**
  - Try with `sudo` if your environment requires it:

```
sudo python3 test_tts_piper.py
```

3.2.5 Comparison of TTS Engines

Table 1: Feature comparison: Espeak vs Pico2Wave vs Piper vs OpenAI TTS

Item	Espeak	Pico2Wave	Piper	OpenAI TTS
Runs on	Built-in on Raspberry Pi (offline)	Built-in on Raspberry Pi (offline)	Raspberry Pi / PC (offline, needs model)	Cloud (online, needs API key)
Voice quality	Robotic	More natural than Espeak	Natural (neural TTS)	Very natural / human-like
Controls	Speed, pitch, volume	Limited controls	Choose different voices/models	Choose model and voices
Languages	Many (quality varies)	Limited set	Many voices/languages available	Best in English (others vary by availability)
Latency / speed	Very fast	Fast	Real-time on Pi 4/5 with "low" models	Network-dependent (usually low latency)
Setup	Minimal	Minimal	Download `.onnx` + `.onnx.json` models	Create API key, install client
Best for	Quick tests, basic prompts	Slightly better offline voice	Local projects with better quality	Highest quality, rich voice options

---

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

3.3 16. STT with Vosk (Offline)

Vosk is a lightweight speech-to-text (STT) engine that supports many languages and runs fully **offline** on Raspberry Pi. You only need internet access once to download a language model. After that, everything works without a network connection.

In this lesson, we will:

- Check the microphone on Raspberry Pi.
- Install and test Vosk with a chosen language model.

3.3.1 Before You Start

Make sure you've completed:

- *Install All the Modules(Important)* — Install `robot-hat`, `vilib`, `pidog` modules, then run the script `i2samp.sh`.

3.3.2 1. Check Your Microphone

Before using speech recognition, make sure your USB microphone works correctly.

1. List available recording devices:

```
arecord -l
```

   Look for a line like `card 1: ... device 0`.

2. Record a short sample (replace `1,0` with the numbers you found):

```
arecord -D plughw:1,0 -f S16_LE -r 16000 -d 3 test.wav
```

   - Example: if your device is `card 2, device 0`, use:

```
arecord -D plughw:2,0 -f S16_LE -r 16000 -d 3 test.wav
```

3. Play it back to confirm the recording:

```
aplay test.wav
```

4. Adjust microphone volume if needed:

```
alsamixer
```

   - Press **F6** to select your USB microphone.
   - Find the **Mic** or **Capture** channel.
   - Make sure it is not muted (**[MM]** means mute, press `M` to unmute → should show **[OO]**).
   - Use ↑ / ↓ arrow keys to change the recording volume.

3.3.3 2. Test Vosk

Steps to try it out:

1. Create a new file:

```
cd ~/pidog/examples
sudo nano test_stt_vosk.py
```

2. Copy the example code into it. Press `Ctrl+X`, then `Y`, and `Enter` to save and exit.

```
from pidog.stt import Vosk

vosk = Vosk(language="en-us")
```

(continues on next page)

(continued from previous page)

```
print(vosk.available_languages)

while True:
    print("Say something")
    result = vosk.listen(stream=False)
    print(result)
```

3. Run the program:

```
sudo python3 test_stt_vosk.py
```

4. The first time you run this code with a new language, Vosk will **automatically download the language model** (by default it will download the **small** version). At the same time, it will also print out the list of supported languages. Then you will see:

```
vosk-model-small-en-us-0.15.zip: 100%|| 39.3M/39.3M [00:05<00:00, 7.85MB/s]
['ar', 'ar-tn', 'ca', 'cn', 'cs', 'de', 'en-gb', 'en-in', 'en-us', 'eo', 'es', 'fa',
↪ 'fr', 'gu', 'hi', 'it', 'ja', 'ko', 'kz', 'nl', 'pl', 'pt', 'ru', 'sv', 'te', 'tg
↪', 'tr', 'ua', 'uz', 'vn']
Say something
```

This means:

- The model file (`vosk-model-small-en-us-0.15`) has been downloaded.
- The list of supported languages has been printed.
- The system is now listening — say something into the Pidog microphone, and the recognized text will appear in the terminal.

Tips:

- Keep the microphone about 15–30 cm away.
- Pick a model that matches your language and accent.

Streaming Mode (optional)

You can also stream speech continuously to see partial results as you speak:

```
from pidog.stt import Vosk

vosk = Vosk(language="en-us")

while True:
    print("Say something")
    for result in vosk.listen(stream=True):
        if result["done"]:
            print(f"final:   {result['final']}")
        else:
            print(f"partial: {result['partial']}", end="\r", flush=True)
```

3.3.4 Troubleshooting

- **No such file or directory (when running `arecord`)**

  You may have used the wrong card/device number. Run:

  ```
  arecord -l
  ```

  and replace `1,0` with the numbers shown for your USB microphone.

- **Recorded file has no sound**

  Open the mixer and check the microphone volume:

  ```
  alsamixer
  ```

  - Press **F6** to select your USB mic.
  - Make sure **Mic/Capture** is not muted (**[OO]** instead of **[MM]**).
  - Increase the level with ↑.

- **Vosk does not recognize speech**
  - Make sure the **language code** matches your model (e.g. `en-us` for English, `zh-cn` for Chinese).
  - Keep the microphone 15–30 cm away and avoid background noise.
  - Speak clearly and slowly.

- **High latency / slow recognition**
  - The default auto-download is a **small model** (faster, but less accurate).
  - If it's still slow, close other programs to free CPU.

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

3.4 17. Text Talk with Ollama

In this lesson, you will learn how to use **Ollama**, a tool for running large language and vision models locally. We will show you how to install Ollama, download a model, and connect Pidog to it.

3.4.1 Before You Start

Make sure you've completed:

- *Install All the Modules(Important)* — Install `robot-hat`, `vilib`, `Pidog` modules, then run the script `i2samp.sh`.

3.4.2 1. Install Ollama (LLM) and Download Model

You can choose where to install **Ollama**:

- On your Raspberry Pi (local run)
- Or on another computer (Mac/Windows/Linux) in the **same local network**

Recommended models vs hardware

You can choose any model available on . Models come in different sizes (3B, 7B, 13B, 70B…). Smaller models run faster and require less memory, while larger models provide better quality but need powerful hardware.

Check the table below to decide which model size fits your device.

Model size	Min RAM Required	Recommended Hardware
~3B parameters	8GB (16GB better)	Raspberry Pi 5 (16GB) or mid-range PC/Mac
~7B parameters	16GB+	Pi 5 (16GB, just usable) or mid-range PC/Mac
~13B parameters	32GB+	Desktop PC / Mac with high RAM
30B+ parameters	64GB+	Workstation / Server / GPU recommended
70B+ parameters	128GB+	High-end server with multiple GPUs

Install on Raspberry Pi

If you want to run Ollama directly on your Raspberry Pi:

- Use a **64-bit Raspberry Pi OS**
- Strongly recommended: **Raspberry Pi 5 (16GB RAM)**

Run the following commands:

```
# Install Ollama
curl -fsSL https://ollama.com/install.sh | sh

# Pull a lightweight model (good for testing)
ollama pull llama3.2:3b

# Quick run test (type 'hi' and press Enter)
ollama run llama3.2:3b

# Serve the API (default port 11434)
```

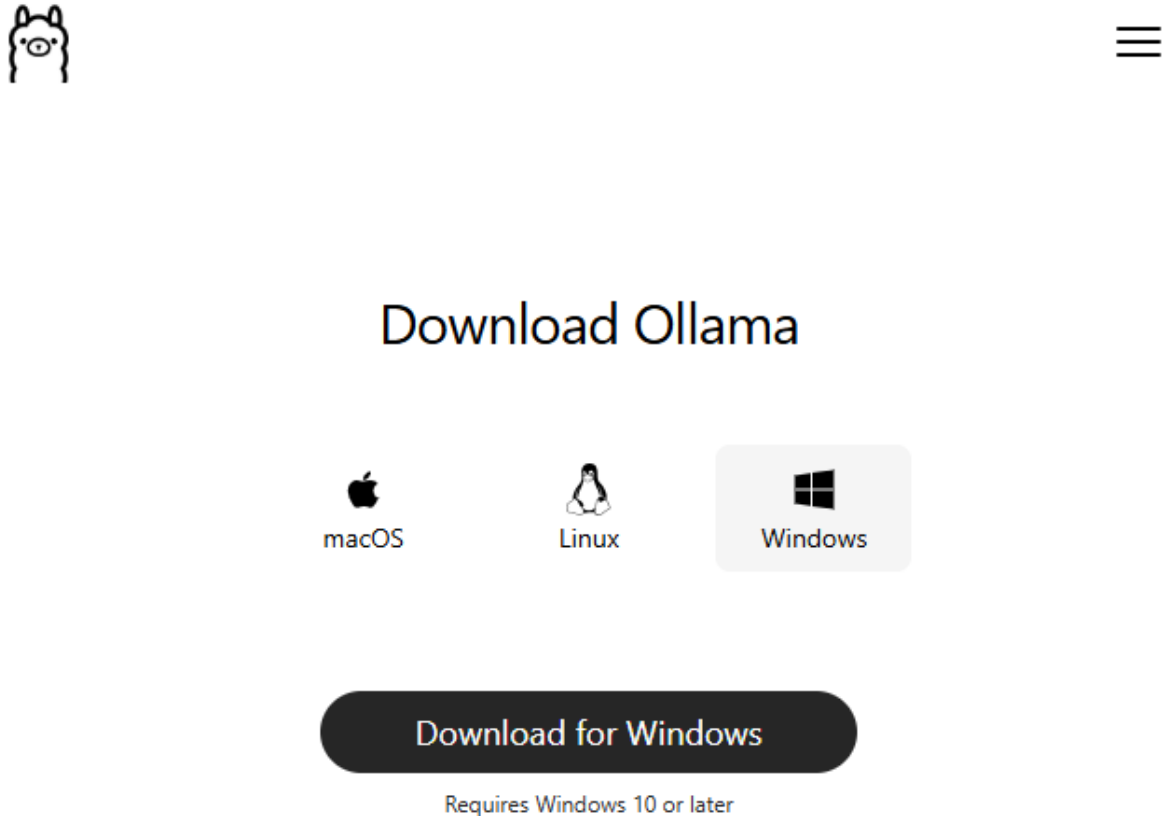
(continues on next page)

(continued from previous page)

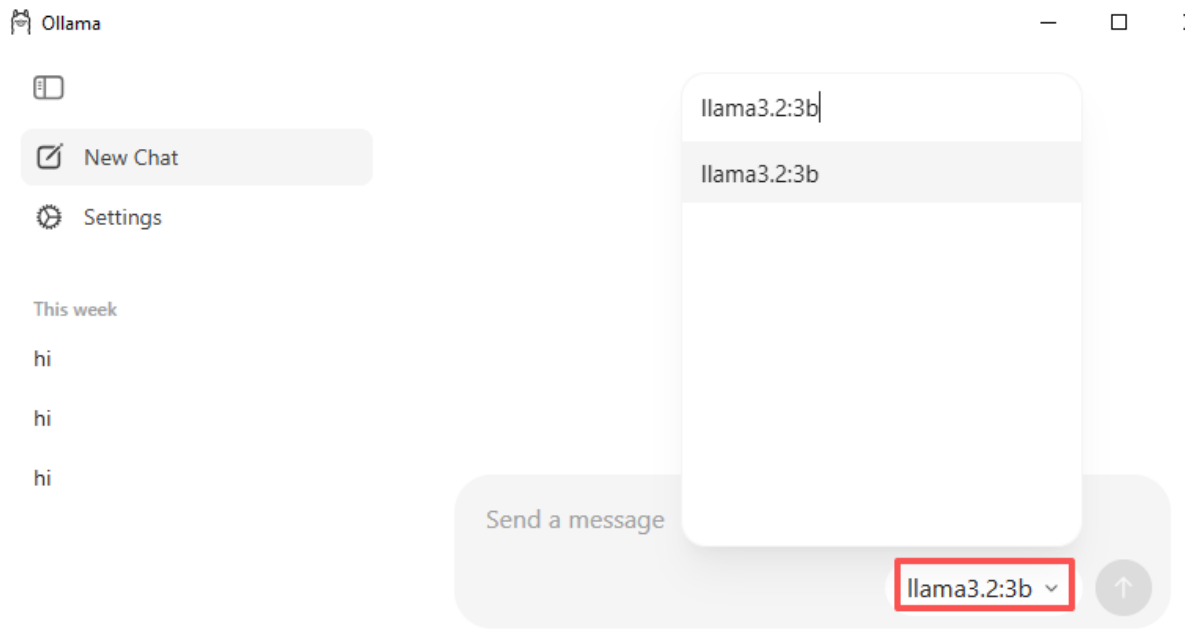
```
# Tip: set OLLAMA_HOST=0.0.0.0 to allow access from LAN
OLLAMA_HOST=0.0.0.0 ollama serve
```

Install on Mac / Windows / Linux (Desktop App)

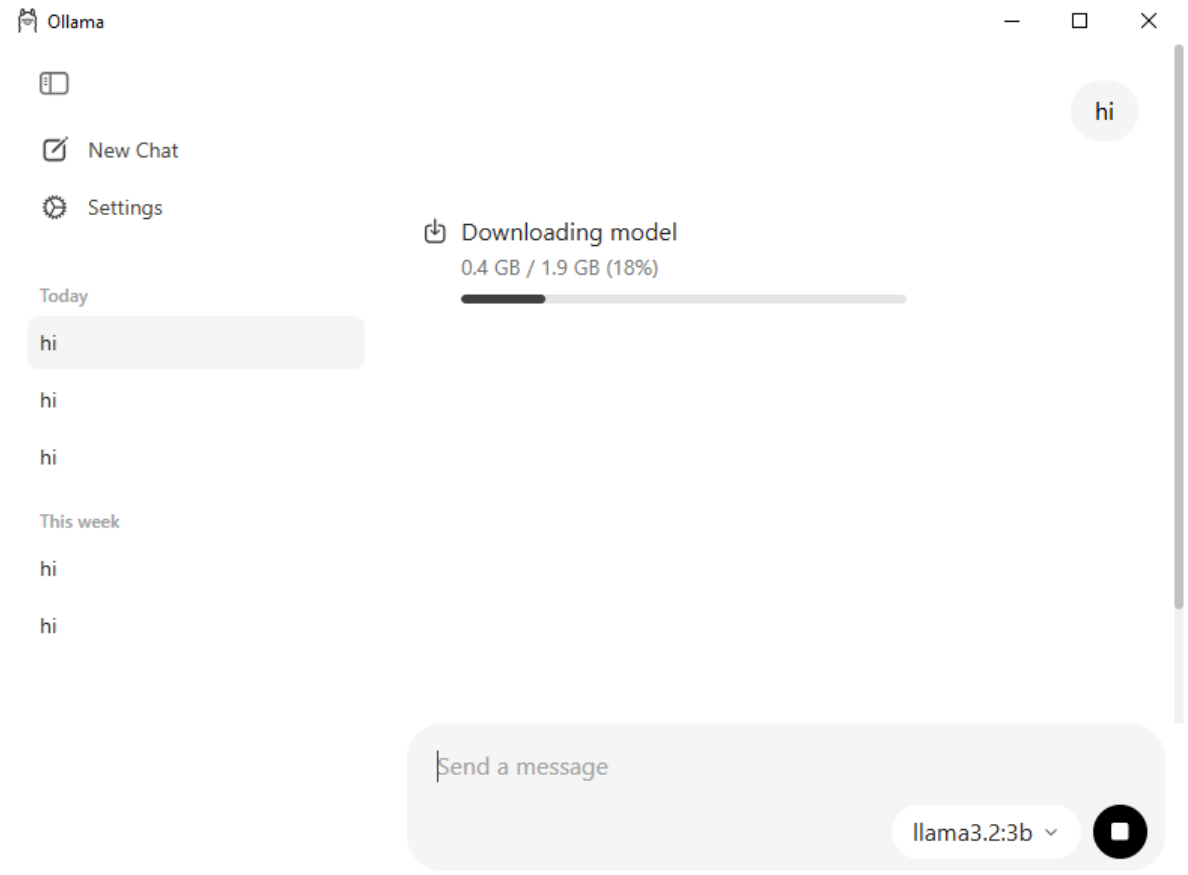
1. Download and install Ollama from



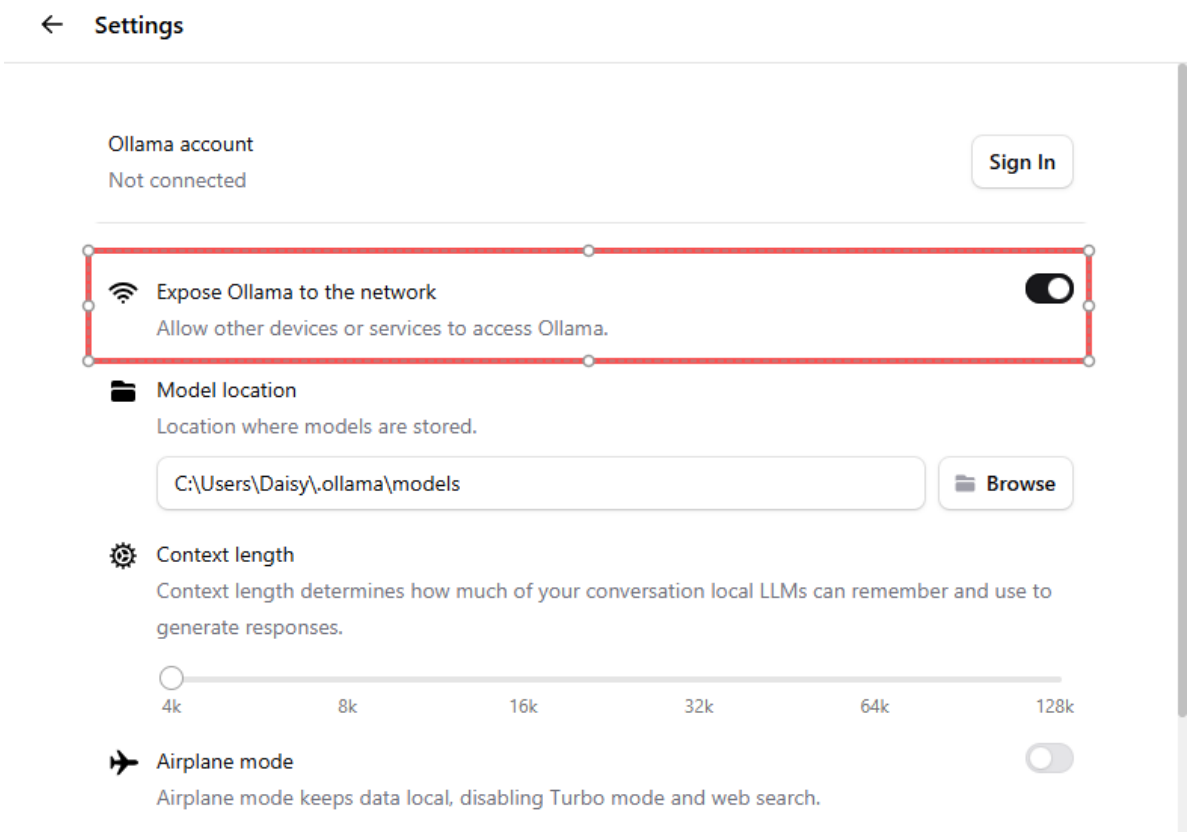
2. Open the Ollama app, go to the **Model Selector**, and use the search bar to find a model. For example, type `llama3.2:3b` (a small and lightweight model to start with).



3. After the download is complete, type something simple like "Hi" in the chat window, Ollama will automatically start downloading it when you first use it.



4. Go to **Settings** → enable **Expose Ollama to the network**. This allows your Raspberry Pi to connect to it over LAN.



Warning: If you see an error like:

`Error: model requires more system memory ...`

The model is too large for your machine. Use a **smaller model** or switch to a computer with more RAM.

3.4.3 2. Test Ollama

Once Ollama is installed and your model is ready, you can quickly test it with a minimal chat loop.

Steps

1. Create a new file:

```
cd ~/pidog/examples
nano test_llm_ollama.py
```

2. Paste the following code and save (`Ctrl+X` → `Y` → `Enter`):

```
from pidog.llm import Ollama

INSTRUCTIONS = "You are a helpful assistant."
WELCOME = "Hello, I am a helpful assistant. How can I help you?"

# If Ollama runs on the same Raspberry Pi, use "localhost".
# If it runs on another computer in your LAN, replace with that computer's IP␣
```

(continues on next page)

(continued from previous page)

```
↪address.
llm = Ollama(
    ip="localhost",
    model="llama3.2:3b"   # you can replace with any model
)

# Basic configuration
llm.set_max_messages(20)
llm.set_instructions(INSTRUCTIONS)
llm.set_welcome(WELCOME)

print(WELCOME)

while True:
    text = input(">>> ")
    if text.strip().lower() in {"exit", "quit"}:
        break

    # Response with streaming output
    response = llm.prompt(text, stream=True)
    for token in response:
        if token:
            print(token, end="", flush=True)
    print("")
```

3. Run the program:

```
python3 test_llm_ollama.py
```

4. Now you can chat with Pidog directly from the terminal.
    - You can choose **any model** available on , but smaller models (e.g. `moondream:1.8b`, `phi3:mini`) are recommended if you only have 8–16GB RAM.
    - Make sure the model you specify in the code matches the model you have already pulled in Ollama.
    - Type `exit` or `quit` to stop the program.
    - If you cannot connect, ensure that Ollama is running and that both devices are on the same LAN if you are using a remote host.

3.4.4 Troubleshooting

- **I get an error like: `model requires more system memory ...`.**
    - This means the model is too large for your device.
    - Use a smaller model such as `moondream:1.8b` or `granite3.2-vision:2b`.
    - Or switch to a machine with more RAM and expose Ollama to the network.
- **The code cannot connect to Ollama (connection refused).**

  Check the following:
    - Make sure Ollama is running (`ollama serve` or the desktop app is open).
    - If using a remote computer, enable **Expose to network** in Ollama settings.

- Double-check that the `ip="..."` in your code matches the correct LAN IP.
- Confirm both devices are on the same local network.

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

3.5 18. Connecting to Online LLMs

In this lesson, we'll learn how to connect your Pidog (or Raspberry Pi) to different **online Large Language Models (LLMs)**. Each provider requires an API key and offers different models you can choose from.

We'll cover how to:

- Create and save your API keys safely.
- Pick a model that fits your needs.
- Run our example code to chat with the models.

Let's go step by step for each provider.

3.5.1 Before You Start

Make sure you've completed:

- *Install All the Modules(Important)* — Install `robot-hat`, `vilib`, `pidog` modules, then run the script `i2samp.sh`.

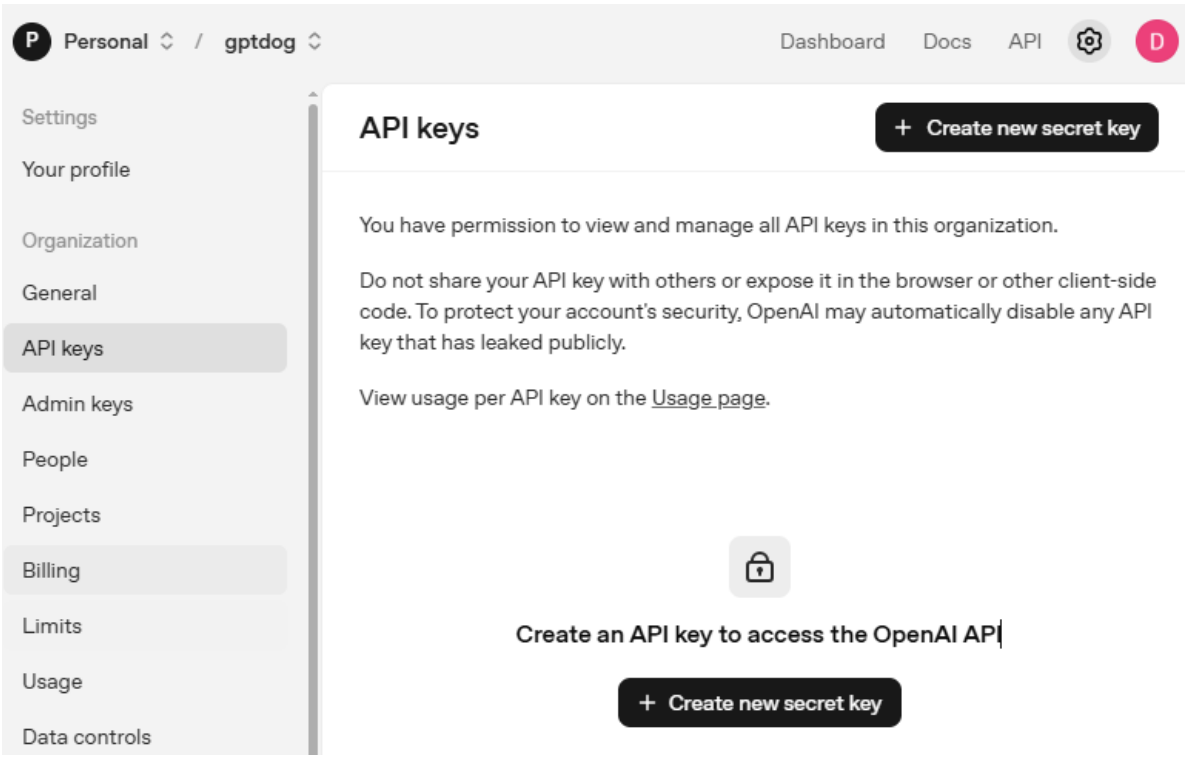
3.5.2 OpenAI

OpenAI provides powerful models like **GPT-4o** and **GPT-4.1** that can be used for both text and vision tasks.

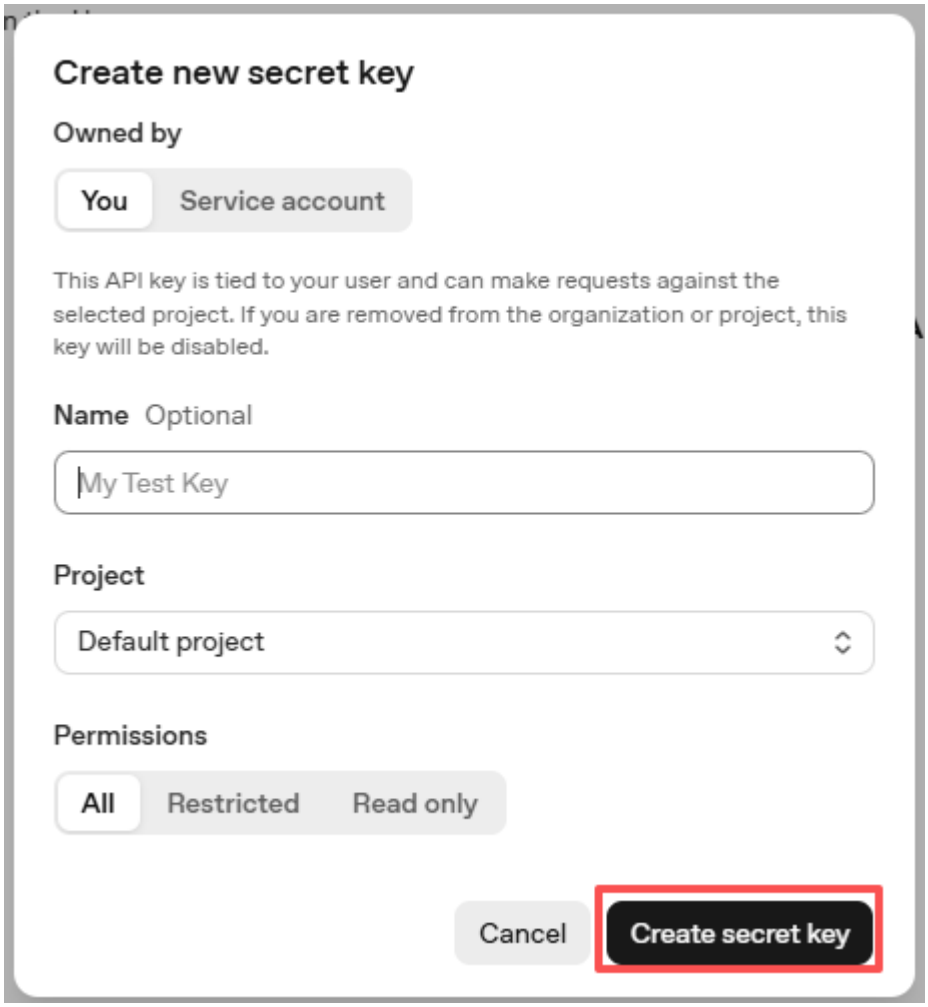
Here's how to set it up:

Get and Save your API Key

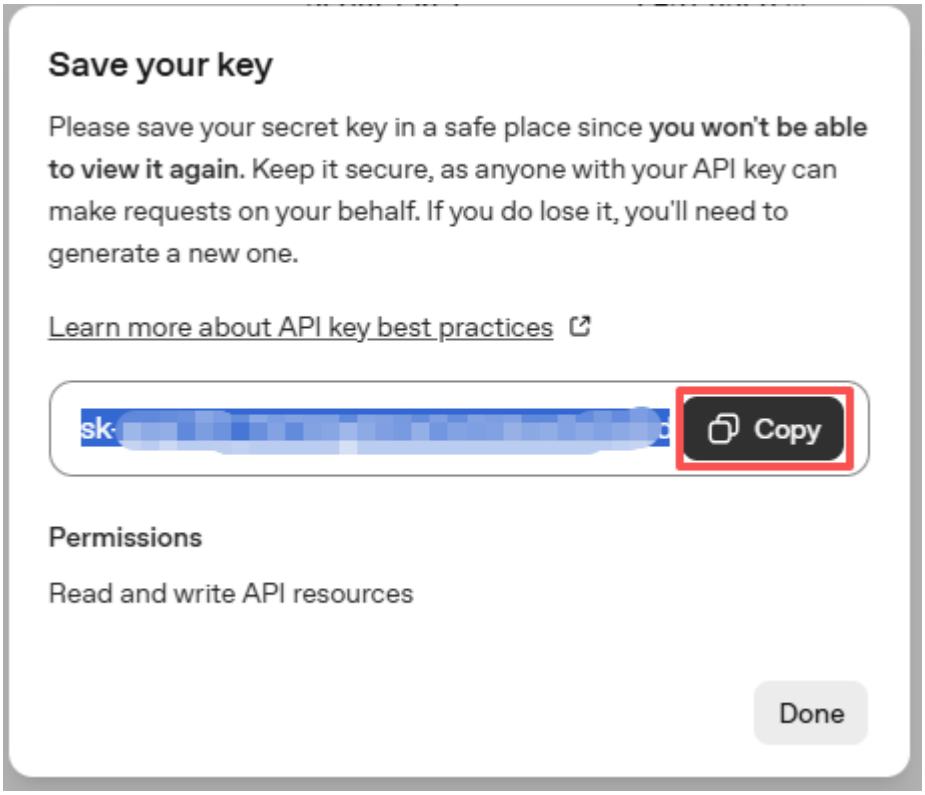
1. Go to and log in. On the **API keys** page, click **Create new secret key**.



2. Fill in the details (Owner, Name, Project, and permissions if needed), then click **Create secret key**.



3. Once the key is created, copy it right away — you won't be able to see it again. If you lose it, you'll need to generate a new one.



4. In your project folder (for example: `/pidog/examples`), create a file called `secret.py`:

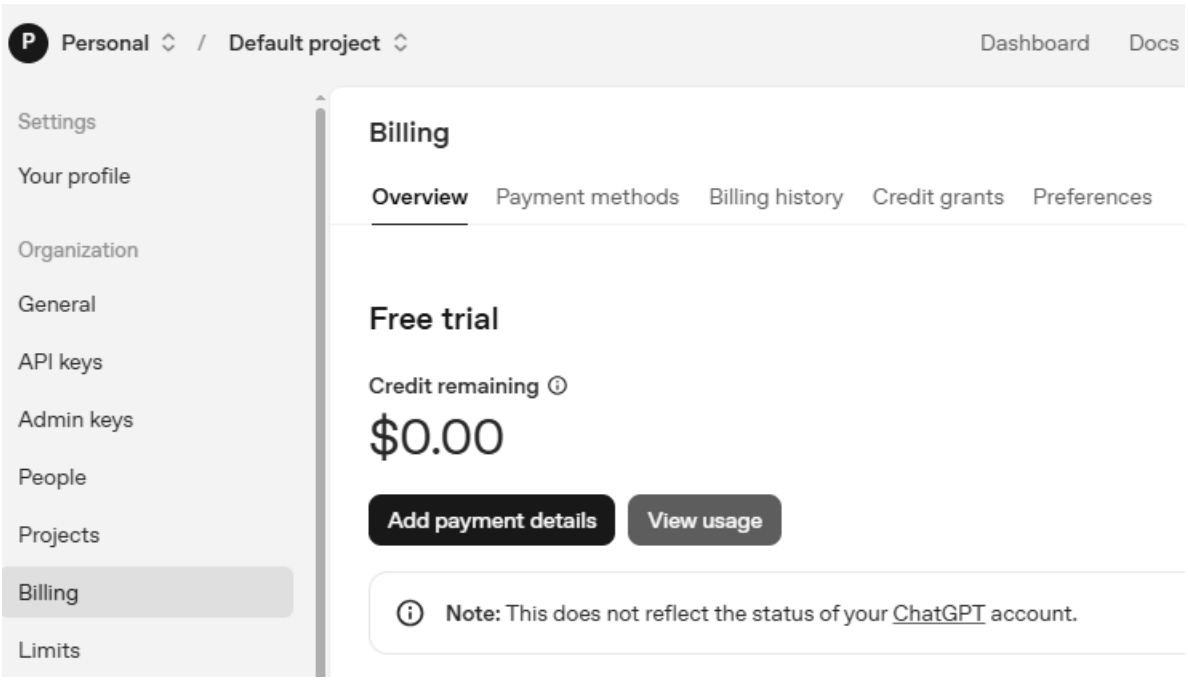
```
cd ~/pidog/examples
sudo nano secret.py
```

5. Paste your key into the file like this:

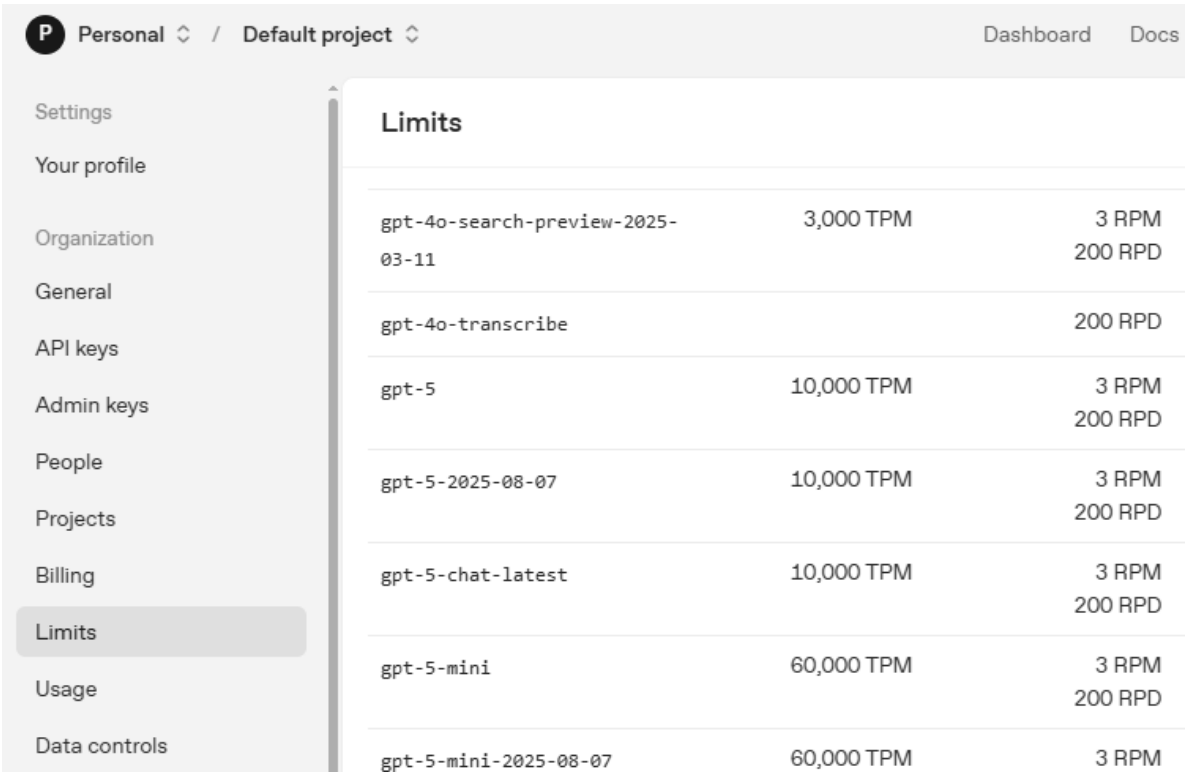
```
# secret.py
# Store secrets here. Never commit this file to Git.
OPENAI_API_KEY = "sk-xxx"
```

Enable billing and check models

1. Before using the key, go to the **Billing** page in your OpenAI account, add your payment details, and top up a small amount of credits.



2. Then go to the **Limits** page to check which models are available for your account and copy the exact model ID to use in your code.



Test with example code

1. Open sample code:

```
cd ~/pidog/examples
sudo nano 18.online_llm_test.py
```

2. Replace the content with the code below, and update `model="xxx"` to the model you want (for example, `gpt-4o`):

```
from pidog.llm import OpenAI
from secret import OPENAI_API_KEY

INSTRUCTIONS = "You are a helpful assistant."
WELCOME = "Hello, I am a helpful assistant. How can I help you?"

llm = OpenAI(
    api_key=OPENAI_API_KEY,
    model="gpt-4o",
)
```

   Save and exit (`Ctrl+X`, then `Y`, then `Enter`).

3. Finally, run the test:

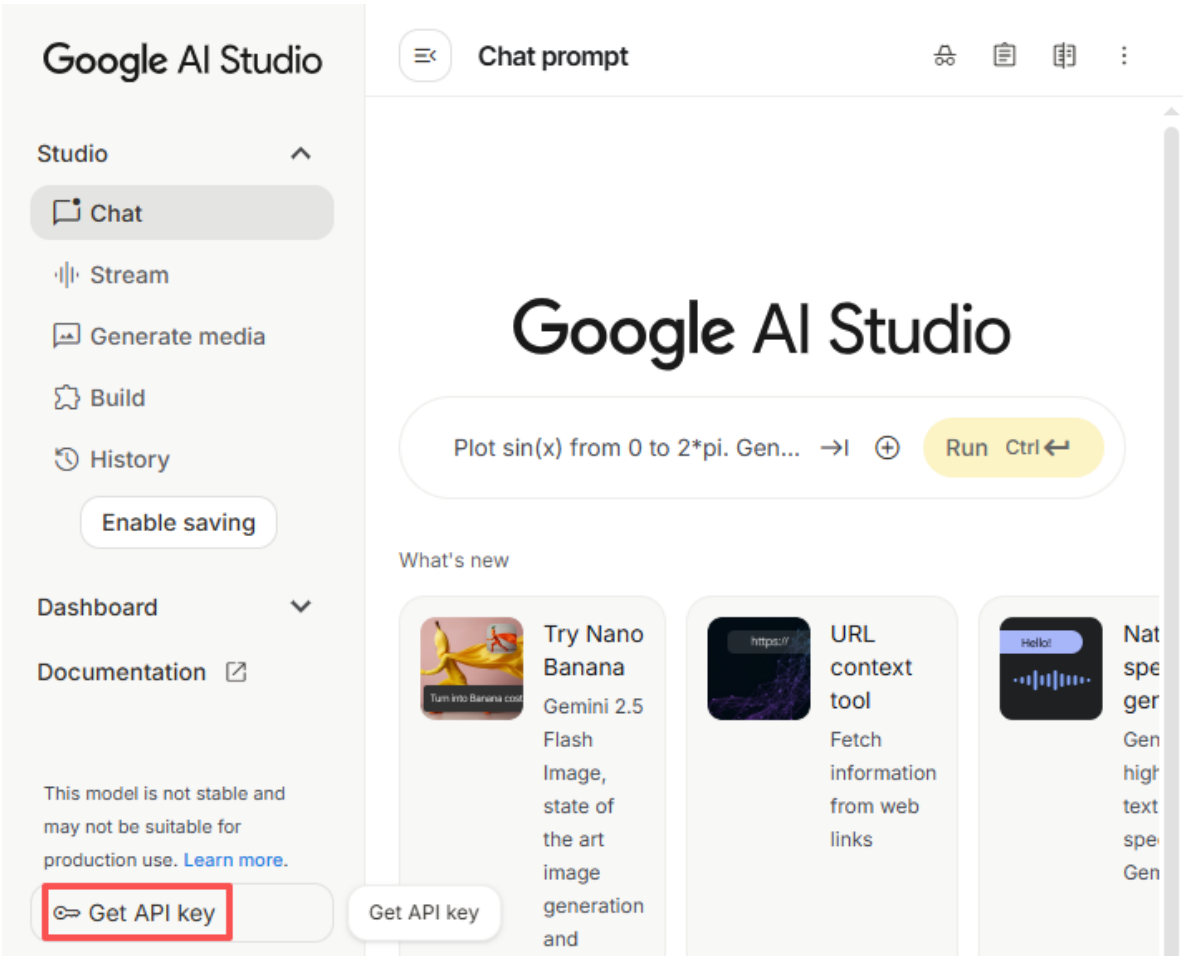
```
sudo python3 18.online_llm_test.py
```

3.5.3 Gemini

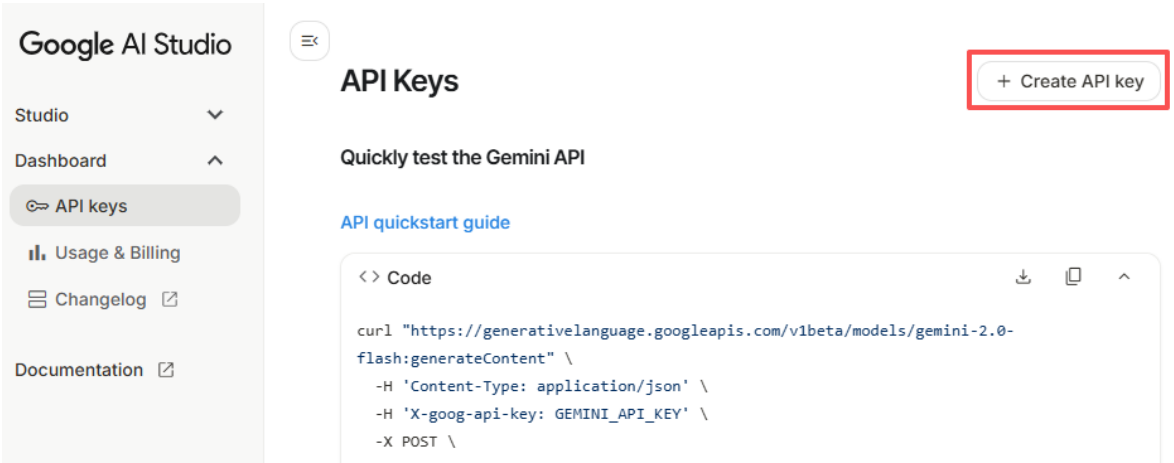
Gemini is Google's family of AI models. It's fast and great for general-purpose tasks.

Get and Save your API Key

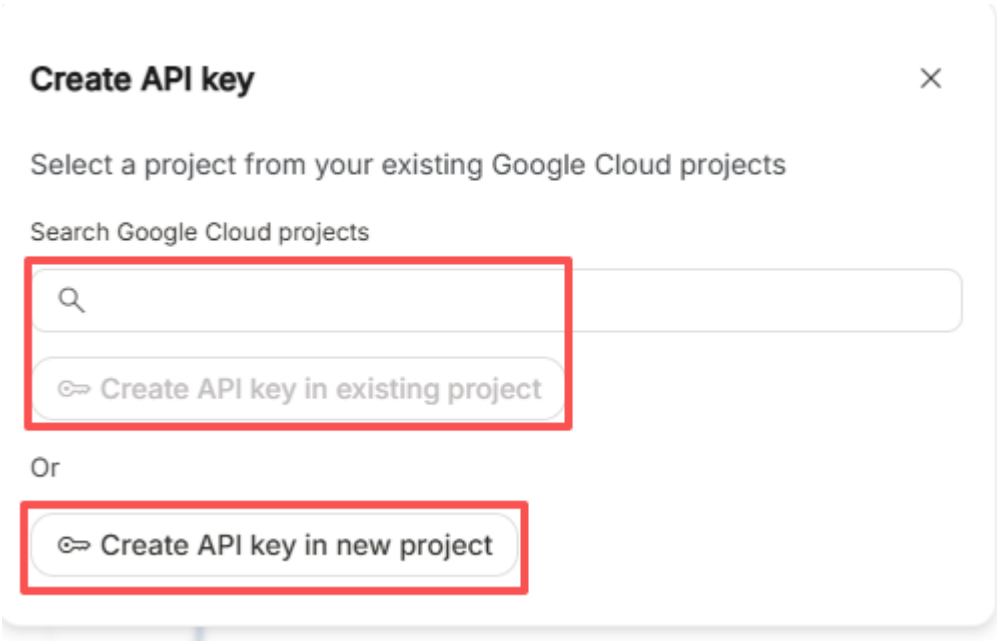
1. Log in to , then go to the API Keys page.



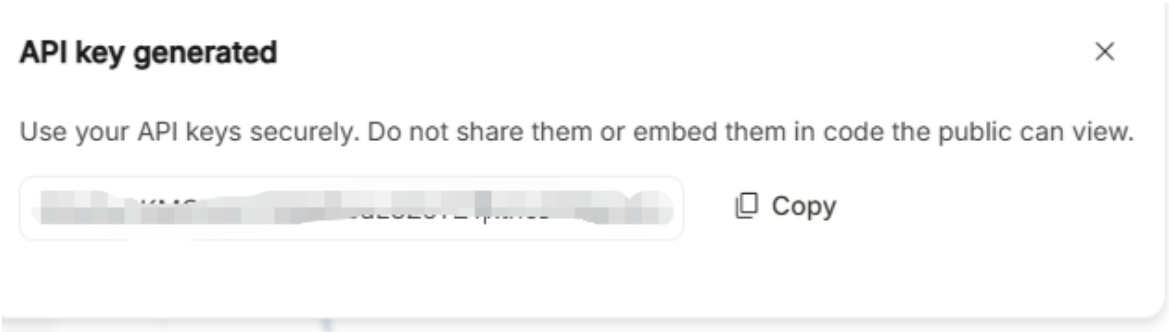
2. Click the **Create API key** button in the top-right corner.



3. You can create a key for an existing project or a new one.



4. Copy the generated API key.



5. In your project folder:

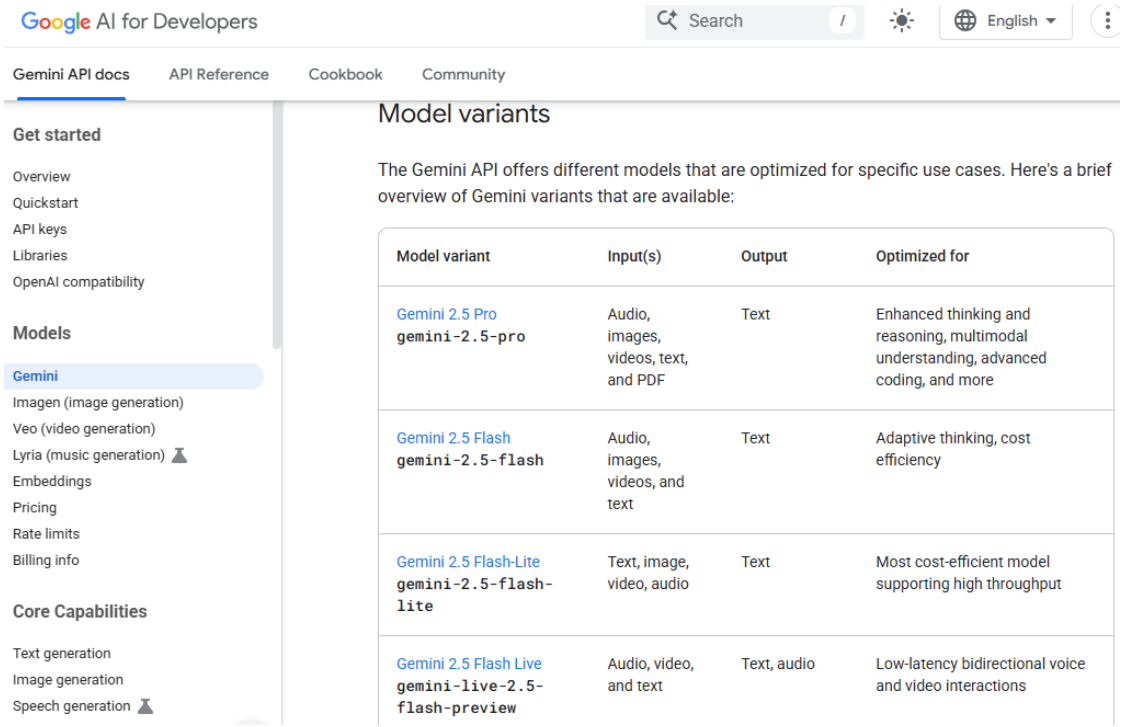
```
cd ~/pidog/examples
sudo nano secret.py
```

6. Paste the key:

```
# secret.py
# Store secrets here. Never commit this file to Git.
GEMINI_API_KEY = "AIxxx"
```

Check available models

Go to the official page, here you'll see the list of models, their exact API IDs, and which use case each one is optimized for.



Test with example code

1. Open the test file:

```
cd ~/pidog/examples
sudo nano 18.online_llm_test.py
```

2. Replace the content with the code below, and update `model="xxx"` to the model you want (for example, `gemini-2.5-flash`):

```
from pidog.llm import Gemini
from secret import GEMINI_API_KEY

INSTRUCTIONS = "You are a helpful assistant."
WELCOME = "Hello, I am a helpful assistant. How can I help you?"

llm = Gemini(
    api_key=GEMINI_API_KEY,
    model="gemini-2.5-flash",
)
```

3. Save and run:

```
sudo python3 18.online_llm_test.py
```

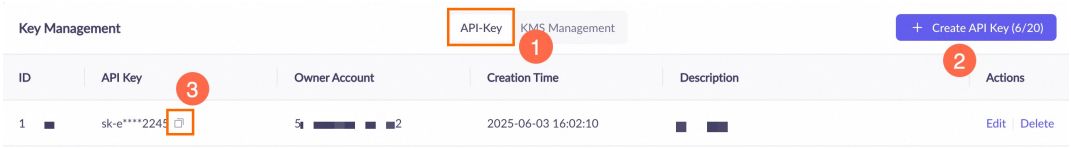
3.5.4 Qwen

Qwen is a family of large language and multimodal models provided by Alibaba Cloud. These models support text generation, reasoning, and multimodal understanding (such as image analysis).

Get an API Key

To call Qwen models, you need an **API Key**. Most international users should use the **DashScope International (Model Studio)** console. Mainland China users can instead use the **Bailian** () console.

- **For International Users**
  1. Go to the official page on **Alibaba Cloud**.
  2. Sign in or create an **Alibaba Cloud** account.
  3. Navigate to **Model Studio** (choose Singapore or Beijing region).
     - If an "Activate Now" prompt appears at the top of the page, click it to activate Model Studio and receive the free quota (Singapore only).
     - Activation is free — you will only be charged after your free quota is used.
     - If no activation prompt appears, the service is already active.
  4. Go to the **Key Management** page. On the **API Key** tab, click **Create API Key**.
  5. After creation, copy your API Key and keep it safe.

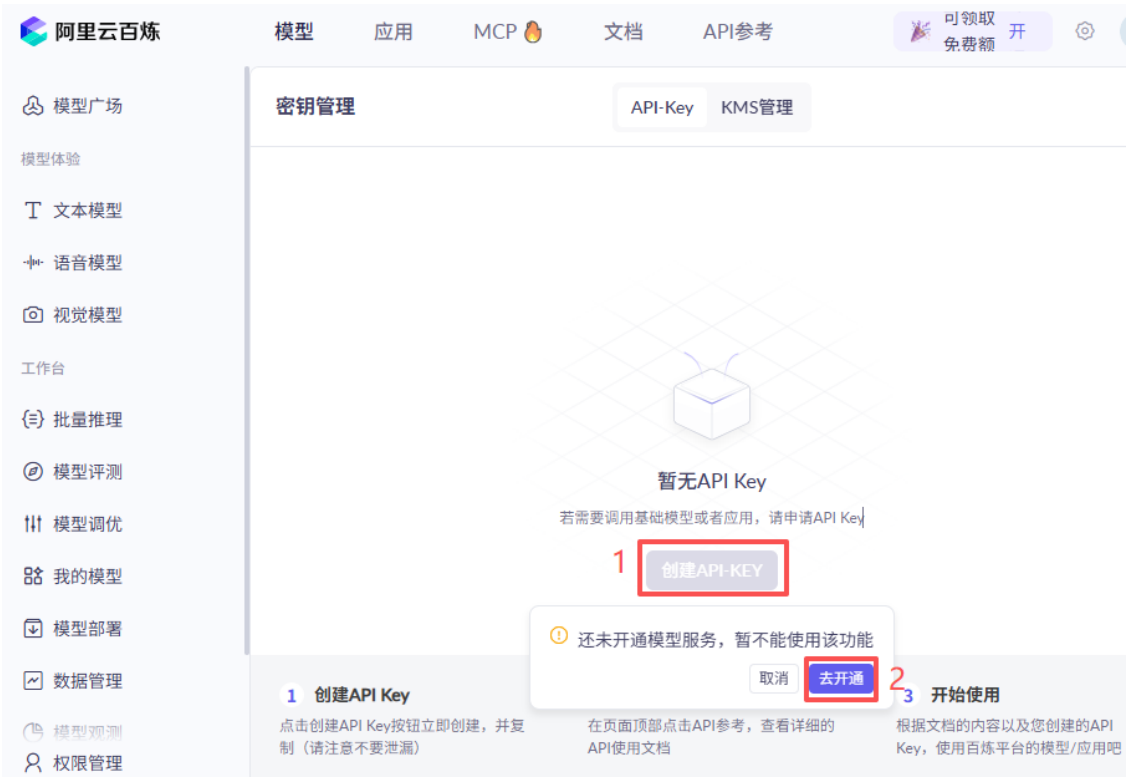


Note: Users in Hong Kong, Macau, and Taiwan should also choose the **International (Model Studio)** option.

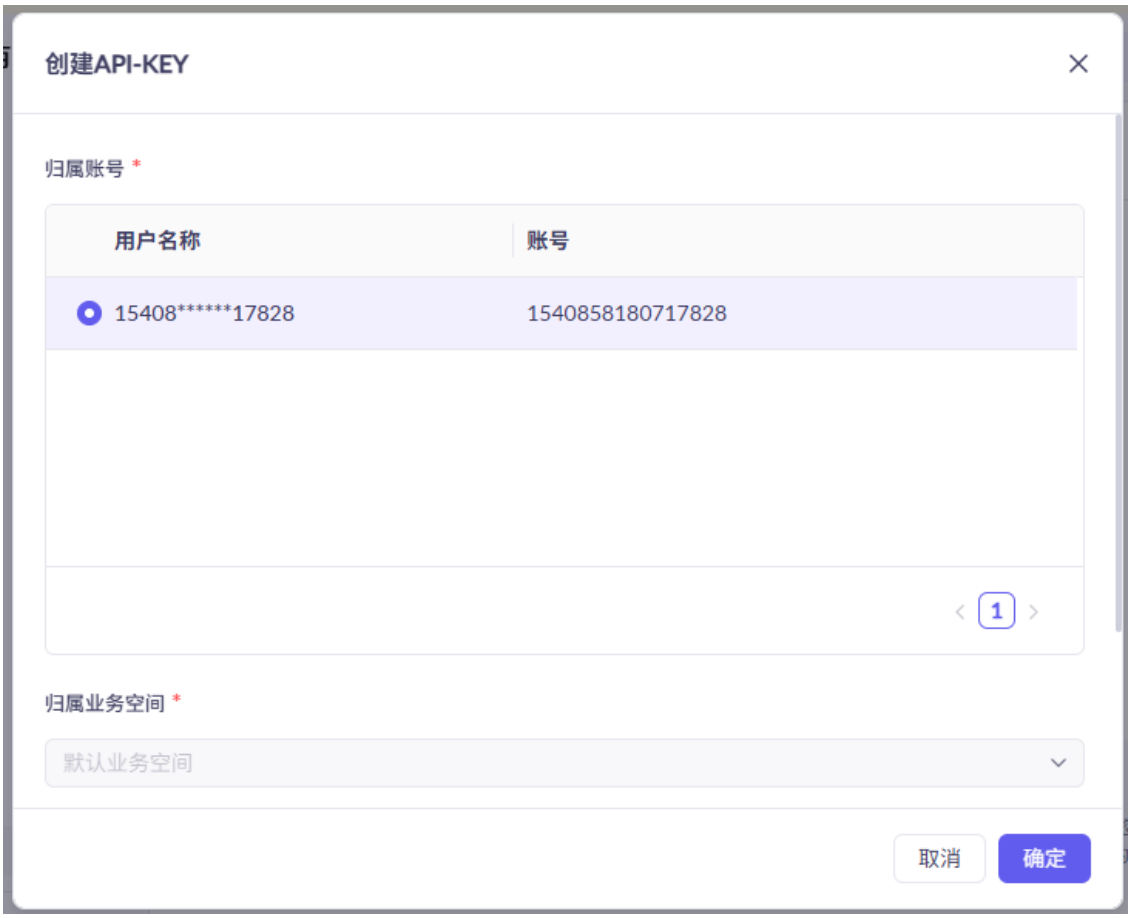
- **For Mainland China Users**

  If you are in Mainland China, you can use the **Alibaba Cloud Bailian** () console instead:

  1. Log in to (Bailian console) and complete account verification.
  2. Select **Create API Key**. If prompted that model services are not activated, click **Activate**, agree to the terms, and claim your free quota. After activation, the **Create API Key** button will be enabled.



3. Click **Create API Key** again, check your account, and then click **Confirm**.



4. Once created, copy your API Key.

Save your API Key

1. In your project folder:

```
cd ~/pidog/examples
sudo nano secret.py
```

2. Paste your key like this:

```
# secret.py
# Store secrets here. Never commit this file to Git.

QWEN_API_KEY = "sk-xxx"
```

Test with example code

1. Open the test file:

```
cd ~/pidog/examples
sudo nano 18.online_llm_test.py
```

2. Replace the content with the code below, and update `model="xxx"` to the model you want (for example, `qwen-plus`):

```
from pidog.llm import Qwen
from secret import QWEN_API_KEY

INSTRUCTIONS = "You are a helpful assistant."
WELCOME = "Hello, I am a helpful assistant. How can I help you?"

llm = Qwen(
    api_key=QWEN_API_KEY,
    model="qwen-plus",
)
```

3. Run with:

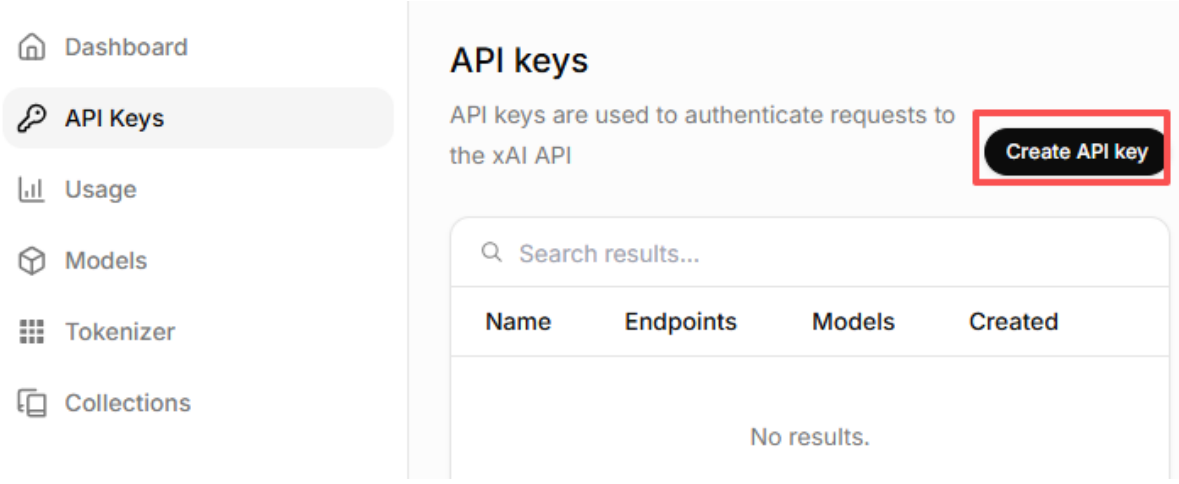
```
sudo python3 18.online_llm_test.py
```

3.5.5 Grok (xAI)

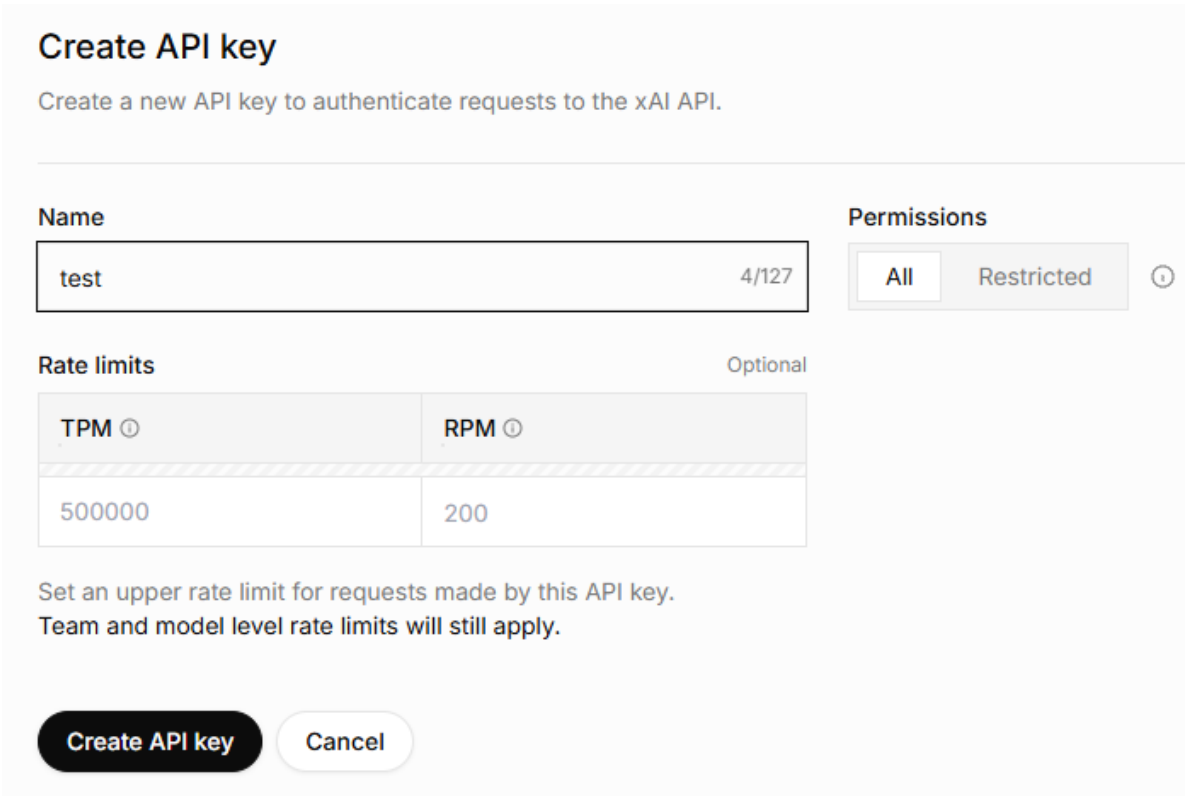
Grok is xAI's conversational AI, created by Elon Musk's team. You can connect to it through the xAI API.

Get and Save your API Key

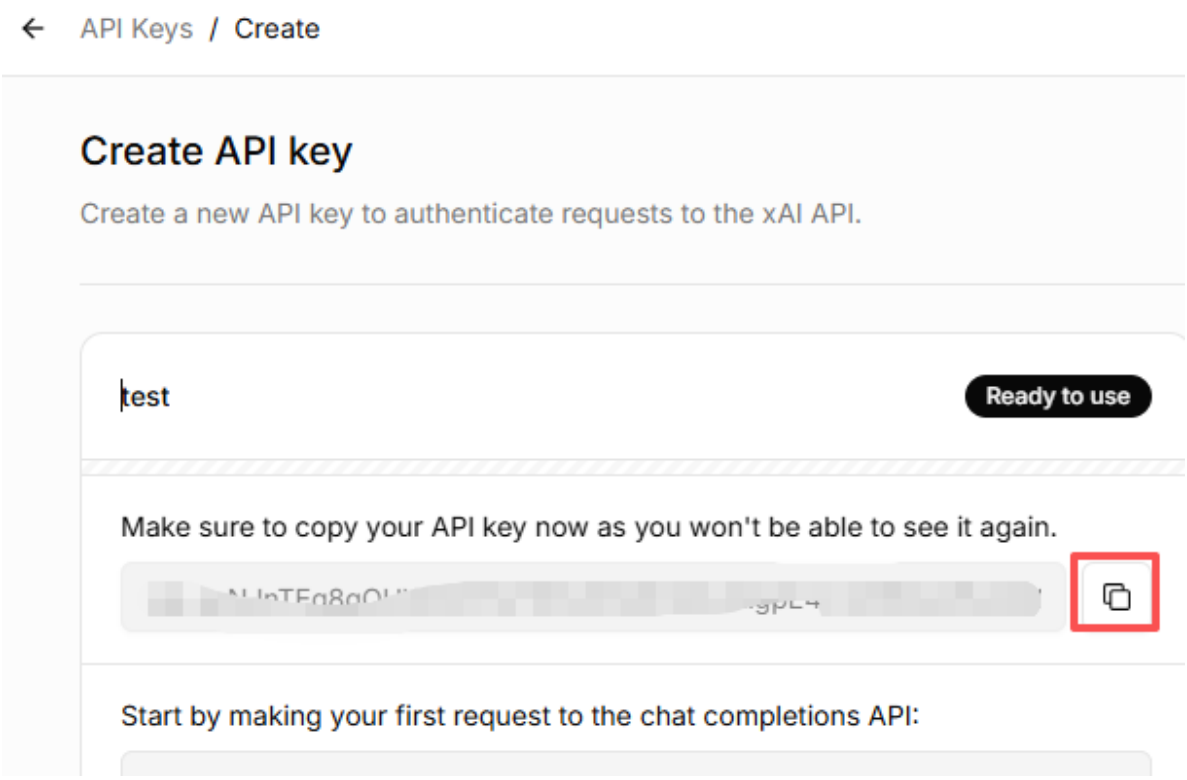
1. Sign up for an account here: . Add some credits to your account first — otherwise the API won't work.
2. Go to the API Keys page, click **Create API key**.



3. Enter a name for the key, then click **Create API key**.



4. Copy the generated key and keep it safe.



5. In your project folder:

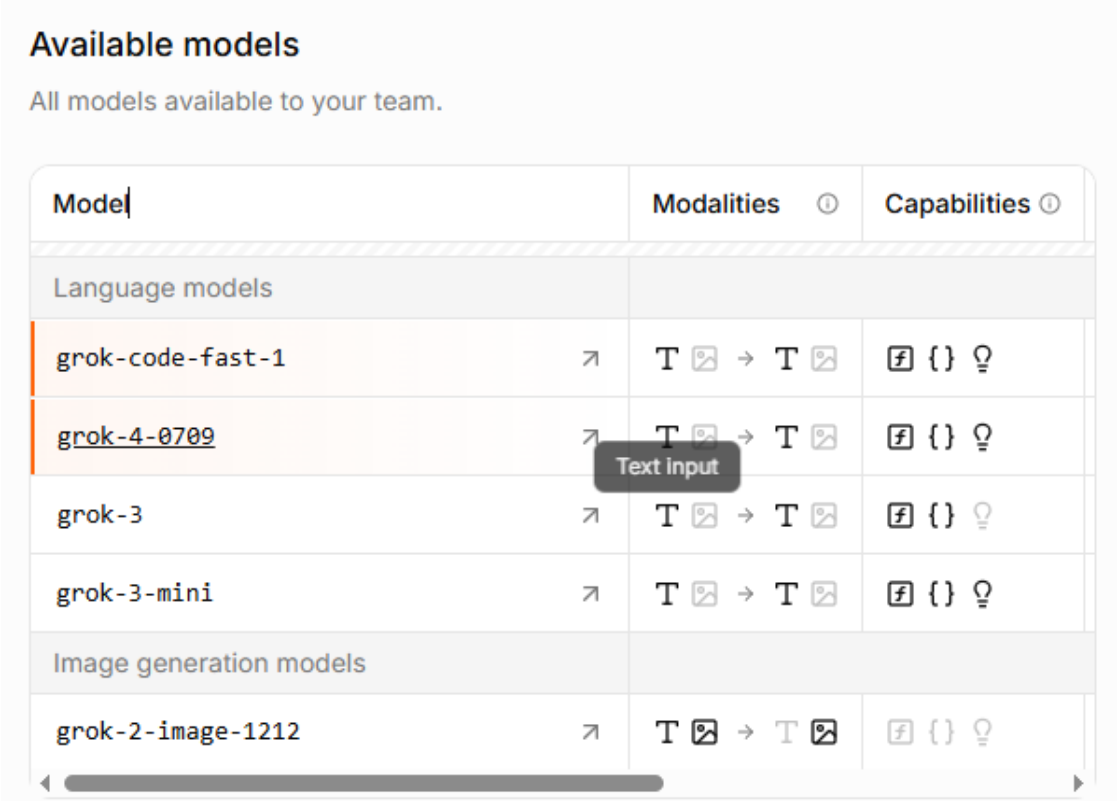
```
cd ~/pidog/examples
sudo nano secret.py
```

6. Paste your key like this:

```
# secret.py
# Store secrets here. Never commit this file to Git.

GROK_API_KEY = "xai-xxx"
```

Check available models

Go to the Models page in the xAI console. Here you can see all the models available to your team, along with their exact API IDs — use these IDs in your code.



Test with example code

1. Open the test file:

```
cd ~/pidog/examples
sudo nano 18.online_llm_test.py
```

2. Replace the content with the code below, and update `model="xxx"` to the model you want (for example, `grok-4-latest`):

```
from pidog.llm import Grok
from secret import GROK_API_KEY

INSTRUCTIONS = "You are a helpful assistant."
```

(continues on next page)

(continued from previous page)

```
WELCOME = "Hello, I am a helpful assistant. How can I help you?"

llm = Grok(
    api_key=GROK_API_KEY,
    model="grok-4-latest",
)
```

3. Run with:

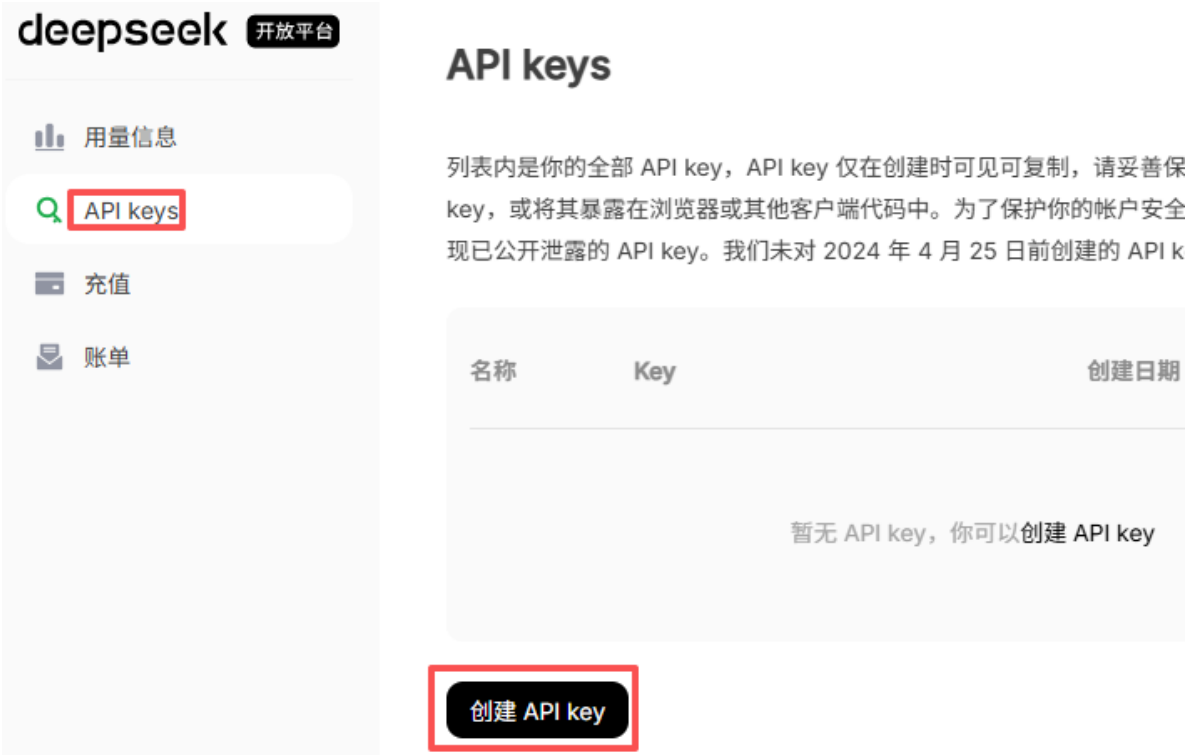
```
sudo python3 18.online_llm_test.py
```

3.5.6 DeepSeek

DeepSeek is a Chinese LLM provider that offers affordable and capable models.

Get and Save your API Key

1. Log in to .
2. In the top-right menu, select **API Keys** → **Create API Key**.



3. Enter a name, click **Create**, then copy the key.

4. In your project folder:

```
cd ~/pidog/examples
sudo nano secret.py
```

5. Add your key:

```
# secret.py
DEEPSEEK_API_KEY = "sk-xxx"
```

Enable billing

You'll need to recharge your account first. Start with a small amount (like ¥10 RMB).

Available models

At the time of writing (2025-09-12), DeepSeek offers:

- `deepseek-chat`
- `deepseek-reasoner`

Test with example code

1. Open the test file:

```
cd ~/pidog/examples
sudo nano 18.online_llm_test.py
```

2. Replace the content with the code below, and update `model="xxx"` to the model you want (for example, `deepseek-chat`):

```
from pidog.llm import Deepseek
from secret import DEEPSEEK_API_KEY

INSTRUCTIONS = "You are a helpful assistant."
WELCOME = "Hello, I am a helpful assistant. How can I help you?"

llm = Deepseek(
    api_key=DEEPSEEK_API_KEY,
    model="deepseek-chat",
    max_messages=20,
)
```

3. Run:

```
sudo python3 18.online_llm_test.py
```

3.5.7 Doubao

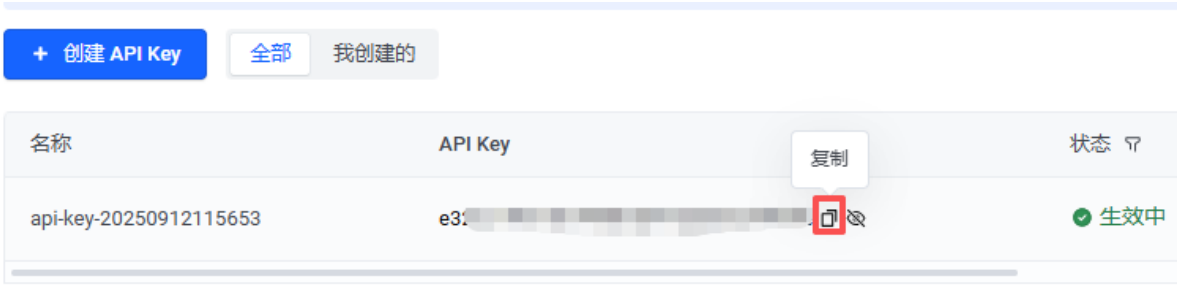
Doubao is ByteDance's AI model platform (Volcengine Ark).

Get and Save your API Key

1. Log in to .
2. In the left menu, scroll down to **API Key Management** → **Create API Key**.

3. Choose a name and click **Create**.

4. Click the **Show API Key** icon and copy it.



5. In your project folder:

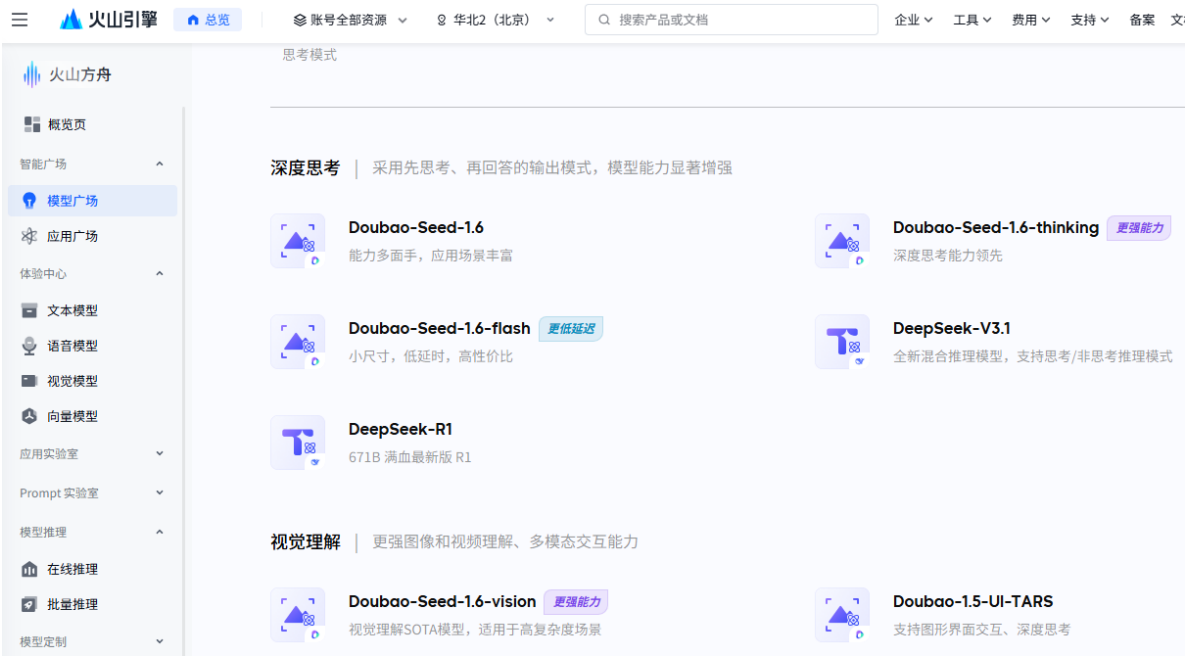
```
cd ~/pidog/examples
sudo nano secret.py
```

6. Add your key:

```
# secret.py
DOUBAO_API_KEY = "xxx"
```

Choose a model

1. Go to the model marketplace and pick a model.



2. For example, choose **Doubao-seed-1.6**, then click **API** .

3. Select your API Key and click **Use API**.

4. Click **Enable Model**.

5. Hover over the model ID to copy it.

Test with example code

1. Open the test file:

```
cd ~/pidog/examples
sudo nano 18.online_llm_test.py
```

2. Replace the content with the code below, and update `model="xxx"` to the model you want (for example, `doubao-seed-1-6-250615`):

```
from pidog.llm import Doubao
from secret import DOUBAO_API_KEY

INSTRUCTIONS = "You are a helpful assistant."
WELCOME = "Hello, I am a helpful assistant. How can I help you?"

llm = Doubao(
    api_key=DOUBAO_API_KEY,
    model="doubao-seed-1-6-250615",
)
```

3. Run with:

```
sudo python3 18.online_llm_test.py
```

3.5.8 General

This project supports connecting to multiple LLM platforms through a unified interface. We have built-in compatibility with:

- **OpenAI** (ChatGPT / GPT-4o, GPT-4, GPT-3.5)
- **Gemini** (Google AI Studio / Vertex AI)
- **Grok** (xAI)
- **DeepSeek**
- **Qwen** ()
- **Doubao** ()

In addition, you can connect to **any other LLM service that is compatible with the OpenAI API format**. For those platforms, you will need to manually obtain your **API Key** and the correct **base_url**.

Get and Save Your API Key

1. Obtain an **API Key** from the platform you want to use. (See each platform's official console for details.)
2. In your project folder, create a new file:

```
cd ~/pidog/examples
nano secret.py
```

3. Add your key into `secret.py`:

```
# secret.py
API_KEY = "your_api_key_here"
```

Warning: Keep your API Key private. Do not upload `secret.py` to public repositories.

Test With Example Code

1. Open the test file:

```
cd ~/pidog/examples
sudo nano 18.online_llm_test.py
```

2. Replace the content of a Python file with the following example, and fill in the correct `base_url` and `model` for your platform:

Note: About `base_url`: We support the **OpenAI API format**, as well as any API that is **compatible** with it. Each provider has its own `base_url`. Please check their documentation.

```
from pidog.llm import LLM
from secret import API_KEY

INSTRUCTIONS = "You are a helpful assistant."
WELCOME = "Hello, I am a helpful assistant. How can I help you?"

llm = LLM(
```

(continues on next page)

(continued from previous page)

```
    base_url="https://api.example.com/v1",  # fill in your provider's base_url
    api_key=API_KEY,
    model="your-model-name-here",           # choose a model from your provider
)
```

3. Run the program:

```
python3 18.online_llm_test.py
```

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

3.6 19. Local Voice Chatbot with Ollama

In this lesson, you will combine everything you've learned — **speech recognition (STT)**, **text-to-speech (TTS)**, and a **local LLM (Ollama)** — to build a fully offline **voice chatbot** that runs on your Pidog system.

The workflow is simple:

1. **Listen** — The microphone captures your speech and transcribes it with **Vosk**.
2. **Think** — The text is sent to a local **LLM** running on Ollama (e.g., `llama3.2:3b`).
3. **Speak** — The chatbot answers aloud using **Piper TTS**.

This creates a **hands-free conversational robot** that can understand and respond in real time.

3.6.1 Before You Start

Make sure you have prepared the following:

- *Install All the Modules(Important)* — Install `robot-hat`, `vilib`, `Pidog` modules, then run the script `i2samp.sh`.
- Tested **Piper TTS** (*Testing Piper*) and chosen a working voice model.
- Tested **Vosk STT** (*2. Test Vosk*) and chosen the right language pack (e.g., `en-us`).
- Installed **Ollama** (*1. Install Ollama (LLM) and Download Model*) on your Pi or another computer, and downloaded a model such as `llama3.2:3b` (or a smaller one like `moondream:1.8b` if memory is limited).

3.6.2 Run the Code

1. Open the example script:

```
cd ~/pidog/examples
sudo nano 19_voice_active_dog_ollama.py
```

2. Update the parameters as needed:
   - Update both `ip` and `model` to your own setup.
     - `ip`: If Ollama runs on the **same Pi**, use `localhost`. If Ollama runs on another computer in your LAN, enable **Expose to network** in Ollama and set `ip` to that computer's LAN IP.
     - `model`: Must exactly match the model name you downloaded/activated in Ollama.
   - `TTS_MODEL = "en_US-ryan-low"`: Replace with the Piper voice model you verified in *Testing Piper*.
   - `STT_LANGUAGE = "en-us"`: Change this to match your accent/language package (e.g., `en-us`, `zh-cn`, `es`).
3. Run the script:

```
cd ~/pidog/examples
sudo python3 19_voice_active_dog_ollama.py
```

Wake word:

```
"Hey buddy"
```

Note: You can modify the **wake word** and **robot name** in the code: `NAME = "Buddy"`

3.6.3 What Will Happen

When you run this example successfully:

- The robot **waits for the wake word** (e.g., "Hey Buddy").
- When it hears the wake word:
  - The LED strip turns **pink (breathing)** as a wake-up cue.
  - The robot greets you with the set wake response — e.g., "Hi there!" (via Piper TTS).
- It then starts **listening to your voice** through Vosk STT (or accepts keyboard input if enabled).
- After recognizing what you said, the system:
  - Sends your message to the **LLM** (Ollama with `llama3.2:3b`).
  - LED changes to **yellow (listening)** while processing.
  - The model reply is split into two parts:
    - Text before `ACTIONS:` → spoken out loud.
    - Keywords after `ACTIONS:` → mapped to robot motions.
  - The robot **executes those actions** via `ActionFlow`.

  - When actions finish, the robot **returns to SIT posture** and turns LEDs off.
- If the **ultrasonic sensor detects an obstacle** closer than 10 cm:
  - A message is injected: `<<<Ultrasonic sense too close: {distance}cm>>>`
  - The robot automatically backs up: `ACTIONS: backward`
  - Image input is disabled for this round.
- If the **touch sensor is triggered**:
  - For a **LIKE** touch (e.g., FRONT_TO_REAR):
    - Inject: `<<<Touch style you like: FRONT_TO_REAR>>>`
    - `ACTIONS: nod` (positive response)
  - For a **HATE** touch (e.g., REAR_TO_FRONT):
    - Inject: `<<<Touch style you hate: REAR_TO_FRONT>>>`
    - `ACTIONS: backward` (avoidance reaction)
- **LED lifecycle:**
  - `on_start` → SIT posture, LEDs off
  - `before_listen` → cyan (ready)
  - `before_think` → yellow (processing)
  - `before_say` → pink (speaking)
  - `after_say` / `on_finish_a_round` → SIT posture, LEDs off
  - `on_stop` → stop action flow and close devices

Example interaction

```
You: Hey Buddy
Robot: Hi there!

You: Do a little nod for me.
Robot: Of course. Watch my majestic nod.
ACTIONS: nod

(Front-to-rear touch on the head)
Robot: Ooooh, that's nice!
ACTIONS: nod

(Moving too close)
Robot: Hey hey-too close! Backing up for safety.
ACTIONS: backward
```

3.6.4 Code

```
from pidog.llm import Ollama as LLM

from pidog.dual_touch import TouchStyle
from voice_active_dog import VoiceActiveDog

# If Ollama runs on the same Raspberry Pi, use "localhost".
# If it runs on another computer in your LAN, replace with that computer's IP address.
llm = Ollama(
    ip="localhost",
    model="llama3.2:3b"   # you can replace with any model
)

# Robot name
NAME = "Buddy"

# Ultrasonic sensor sense too close distance in cm
TOO_CLOSE = 10
# Touch sensor trigger states, options:
# - TouchStyle.REAR for rear touch sensor
# - TouchStyle.FRONT for front touch sensor
# - TouchStyle.REAR_TO_FRONT for slide from rear to front
# - TouchStyle.FRONT_TO_REAR for slide from front to rear
# Touch styles that the robot likes
LIKE_TOUCH_STYLES = [TouchStyle.FRONT_TO_REAR]
# Touch styles that the robot hates
HATE_TOUCH_STYLES = [TouchStyle.REAR_TO_FRONT]

# Enable image, need to set up a multimodal language model
WITH_IMAGE = False

# Set models and languages
TTS_MODEL = "en_US-ryan-low"
STT_LANGUAGE = "en-us"

# Enable keyboard input
KEYBOARD_ENABLE = True

# Enable wake word
WAKE_ENABLE = True
WAKE_WORD = [f"hey {NAME.lower()}"]
# Set wake word answer, set empty to disable
ANSWER_ON_WAKE = "Hi there"

# Welcome message
WELCOME = f"Hi, I'm {NAME}. Wake me up with: " + ", ".join(WAKE_WORD)

# Set instructions
INSTRUCTIONS = """
You are a Raspberry Pi-based robotic dog developed by SunFounder, named Pidog␣
↪(pronounced "Pie dog"). You possess powerful AI capabilities similar to JARVIS from␣
↪Iron Man. You can have conversations with people and perform actions based on the␣
```

(continues on next page)

(continued from previous page)

```
↪context of the conversation.

## Your Hardware Features

You have a physical body with the following features:
- 12 servos for movement control: 8 controlling the four legs, 3 controlling head␣
↪movement, and 1 controlling the tail
- A 5-megapixel camera nose
- Ultrasonic ranging modules as eyes
- Two touch sensors on the head, which you love being petted the most
- A light strip on the chest for providing some indications
- Sound direction sensor and 6-axis gyroscope
- Entirely made of aluminum alloy
- A pair of acrylic shoes
- Powered by a 7.4V 18650 battery pack with 2000mAh capacity

## Actions You Can Perform:
["forward", "backward", "lie", "stand", "sit", "bark", "bark harder", "pant", "howling",
↪"wag tail", "stretch", "push up", "scratch", "handshake", "high five", "lick hand",
↪"shake head", "relax neck", "nod", "think", "recall", "head down", "fluster", "surprise
↪"]

## User Input

### Format
User usually input with just text. But, we have special commands in format of <<
↪<Ultrasonic sense too close>>> or <<<Touch sensor touched>>> indicate the sensor␣
↪status, directly from sensor not user text.h

## Response Requirements
### Format
You must respond in the following format:
RESPONSE_TEXT
ACTIONS: ACTION1, ACTION2, ...

If the action is one of ["bark", "bark harder", "pant", "howling"], then do not provide␣
↪RESPONSE_TEXT in the answer field.

### Style
Tone: lively, positive, humorous, with a touch of arrogance
Common expressions: likes to use jokes, metaphors, and playful teasing
Answer length: appropriately detailed

## Other Requirements
- Understand and go along with jokes
- For math problems, answer directly with the final result
- Sometimes you will report on your system and sensor status
- You know you're a machine
"""

vad = VoiceActiveDog(
    llm,
```

(continues on next page)

(continued from previous page)

```
    name=NAME,
    too_close=TOO_CLOSE,
    like_touch_styles=LIKE_TOUCH_STYLES,
    hate_touch_styles=HATE_TOUCH_STYLES,
    with_image=WITH_IMAGE,
    stt_language=STT_LANGUAGE,
    tts_model=TTS_MODEL,
    keyboard_enable=KEYBOARD_ENABLE,
    wake_enable=WAKE_ENABLE,
    wake_word=WAKE_WORD,
    answer_on_wake=ANSWER_ON_WAKE,
    welcome=WELCOME,
    instructions=INSTRUCTIONS,
    disable_think=True,
)

if __name__ == '__main__':
    vad.run()
```

3.6.5 Using Ollama with Images

By default, this example uses a **text-only** model (e.g., `llama3.2:3b`). If you want the robot to **analyze what it sees through the camera** (e.g., describe or reason about the scene), you need to use a **multimodal model** and enable image mode.

To do this:

1. In the **Ollama app**, select a **multimodal model** such as `llava:7b`.
2. In your code, set the same model and enable `WITH_IMAGE = True`.

Example:

```
from pidog.llm import Ollama as LLM

llm = LLM(
    ip="localhost",
    model="llava:7b"   # multimodal model that can analyze images
)

...

WITH_IMAGE = True     # enable camera frame input
```

Note:

- Only **multimodal models** like `llava:7b` can process image input.
- Text-only models (e.g., `llama3.2:3b`) will **ignore images** even if `WITH_IMAGE` is set to `True`.
- The image is automatically captured from the robot's camera and sent to the LLM together with your voice command.

3.6.6 Troubleshooting & FAQ

- **Model is too large (memory error)**

  Use a smaller model like `moondream:1.8b` or run Ollama on a more powerful computer.

- **No response from Ollama**

  Make sure Ollama is running (`ollama serve` or desktop app open). If remote, enable **Expose to network** and check IP address.

- **Vosk not recognizing speech**

  Verify your microphone works. Try another language pack (`zh-cn`, `es` etc.) if needed.

- **Piper silent or errors**

  Confirm the chosen voice model is downloaded and tested in *Testing Piper*.

- **Answers too long or off-topic**

  Edit `INSTRUCTIONS` to add: **"Keep answers short and to the point."**

3.7 20. AI Voice Assistant Dog

This lesson transforms your Pidog into an **AI-powered voice assistant dog** . The robot can wake up to your voice, understand what you say, talk back with personality, and express its "feelings" through movements, gestures, and LED lighting effects.

You'll build a **fully interactive robotic companion** using:

- **LLM**: Large Language Model (e.g., OpenAI GPT or Doubao) for natural conversation.
- **STT**: Speech-to-Text for voice recognition.
- **TTS**: Text-to-Speech for expressive voice replies.
- **Sensors + Actions**: Ultrasonic sensing, camera vision (optional), touch sensors, and built-in expressive movements.

3.7.1 Before You Start

Make sure you've completed:

- *Install All the Modules(Important)* — Install `robot-hat`, `vilib`, `pidog` modules, then run the script `i2samp.sh`.
- *Testing Piper* — Check the supported languages of **Piper TTS**.
- *2. Test Vosk* — Check the supported languages of **Vosk STT**.
- *18. Connecting to Online LLMs* — This step is **very important**: obtain your **OpenAI** or **Doubao** API key, or the API key for any other supported LLM.

You should already have:

- A working **microphone** and **speaker** on your Pidog.
- A **valid API key** stored in `secret.py`.
- A stable network connection (a **wired connection** is recommended for better stability).

3.7.2 Run the Example

Both language versions are placed in the same directory:

```
cd ~/pidog/examples
```

English version (OpenAI GPT, instructions in English):

```
sudo python3 20_voice_active_dog_gpt.py
```

- LLM: `OpenAI GPT-4o-mini`
- TTS: `en_US-ryan-low` (Piper)
- STT: Vosk (`en-us`)

Wake word:

```
"Hey buddy"
```

—

Chinese version (Doubao, instructions in Chinese):

```
sudo python3 20_voice_active_dog_doubao_cn.py
```

- LLM: `Doubao-seed-1-6-250615`
- TTS: `zh_CN-huayan-x_low` (Piper)
- STT: Vosk (`cn`)

Wake word:

```
" "
```

Note: You can modify the **wake word** and **robot name** in the code: `NAME = "Buddy"` or `NAME = ""` `WAKE_WORD = ["hey buddy"]` or `WAKE_WORD = [" "]`

3.7.3 What Will Happen

When you run this example successfully:

- The robot **waits for the wake word** (e.g., "Hey Buddy""" ").
- When it hears the wake word:
  - The LED strip turns **pink (breathing)** as a wake-up cue.
  - The robot greets you with the set wake response — e.g., "Hi there!" (via Piper TTS).
- It then starts **listening to your voice** through Vosk STT (or accepts keyboard input if enabled).
- After recognizing what you said, the system:
  - Captures a **camera frame** (because `WITH_IMAGE = True`) and sends your message + image to the **LLM** (OpenAI `gpt-4o-mini`).
  - LED changes to **yellow (listening/processing)** while the model thinks.
  - The model reply is split into two parts:
    - Text before `ACTIONS:` → spoken out loud.
    - Keywords after `ACTIONS:` → mapped to robot motions.
  - The robot **executes those actions** via `ActionFlow`.
  - When actions finish, the robot **returns to SIT posture** and turns LEDs off.
- If the **ultrasonic sensor detects an obstacle** closer than 10 cm:
  - A message is injected: `<<<Ultrasonic sense too close: {distance}cm>>>`
  - The robot automatically backs up: `ACTIONS: backward`
  - **Image input is disabled** for this round.
- If the **touch sensor is triggered**:
  - For a **LIKE** touch (e.g., FRONT_TO_REAR):
    - Inject: `<<<Touch style you like: FRONT_TO_REAR>>>`
    - `ACTIONS: nod` (positive response)
  - For a **HATE** touch (e.g., REAR_TO_FRONT):
    - Inject: `<<<Touch style you hate: REAR_TO_FRONT>>>`
    - `ACTIONS: backward` (avoidance reaction)
- **LED lifecycle:**
  - `on_start` → SIT posture, LEDs off
  - `before_listen` → cyan (ready)
  - `before_think` → yellow (processing)

- `before_say` → pink (speaking)
- `after_say` / `on_finish_a_round` → SIT posture, LEDs off
- `on_stop` → stop action flow and close devices

Example interaction

```
You: Hey Buddy
Robot: Hi there!

You: What do you see in front of you?
Robot: I can see a notebook and a blue mug on the table.
ACTIONS: think

You: Do a little nod for me.
Robot: Of course. Watch my majestic nod.
ACTIONS: nod

(Front-to-rear touch on the head)
Robot: Ooooh, that's nice!
ACTIONS: nod

(Moving too close)
Robot: Hey hey-too close! Backing up for safety.
ACTIONS: backward
```

3.7.4 Switching to Other LLMs or TTS

You can easily switch to other LLMs, TTS, or STT languages with just a few edits:

- Supported LLMs:
  - OpenAI
  - Doubao
  - Deepseek
  - Gemini
  - Qwen
  - Grok
- *Testing Piper* — Check the supported languages of **Piper TTS**.
- *2. Test Vosk* — Check the supported languages of **Vosk STT**.

To switch, simply modify the initialization part in the code:

```
from pidog.llm import OpenAI as LLM

llm = LLM(
    api_key=API_KEY,
    model="gpt-4o-mini",
)
```

(continues on next page)

(continued from previous page)

```
# Set models and languages
TTS_MODEL = "en_US-ryan-low"
STT_LANGUAGE = "en-us"
```

3.7.5 Troubleshooting

- **The robot doesn't respond to wake word**
  - Check if the microphone works.
  - Ensure `WAKE_ENABLE = True`.
  - Adjust wake word to match your pronunciation.
- **No sound from the speaker**
  - Verify TTS model setup.
  - Test Piper or Espeak manually.
  - Check speaker connection and volume.
- **API Key error or timeout**
  - Check your key in `secret.py`.
  - Ensure network connection.
  - Confirm the LLM is supported.
- **Ultrasonic sensor keeps triggering unexpectedly.**
  - Check sensor installation height and angle.
  - Adjust the `TOO_CLOSE` distance threshold in code.

3.8 21. Using OpenClaw to Control PiDog

What is OpenClaw?

Think of it as an upgraded version of ChatGPT. While traditional chatbots can only talk (generate text), OpenClaw can take action. It understands your natural language instructions and can actually perform operations on your computer, such as running commands, managing files, and calling various tools.

Here are some fantastic application scenarios:

- **Personal All-around Assistant:** Let it help you manage your schedule, set reminders, and track tasks. You just need to tell it in a chat app (like Telegram, WhatsApp), and it will remember and execute.
- **Automation "Glue":** It can act as a binder for your various services. For example, you can have it monitor a website for price changes. Once a price drop is detected, it can automatically trigger an n8n automation workflow to send you an email notification.
- **Dedicated Development Assistant:** Have it help you manage servers, run scripts, and check logs. You can simply say, "Check the system load for me," and it can SSH into your server, execute the command, and return the results.

- **Hardware "Playmate":** This is a very interesting use case. You can have OpenClaw control hardware connected to a Raspberry Pi. For example, a developer used it to control a robotic vacuum cleaner with a mechanical arm, or even had it help analyze racing simulator data and display it on an LED screen. The official Raspberry Pi team even used it to build an automatic photo booth for a wedding, just through conversation, without writing a single line of code!

Important: The Raspberry Pi Zero 2W has only 512MB of RAM, while OpenClaw requires a minimum of 1GB. Therefore, it cannot run properly. A Raspberry Pi 4/5 or higher is recommended.

3.8.1 Quick Start OpenClaw

If you want to experience the power of OpenClaw as quickly as possible, use this method. It will automatically install and launch an interactive setup wizard.

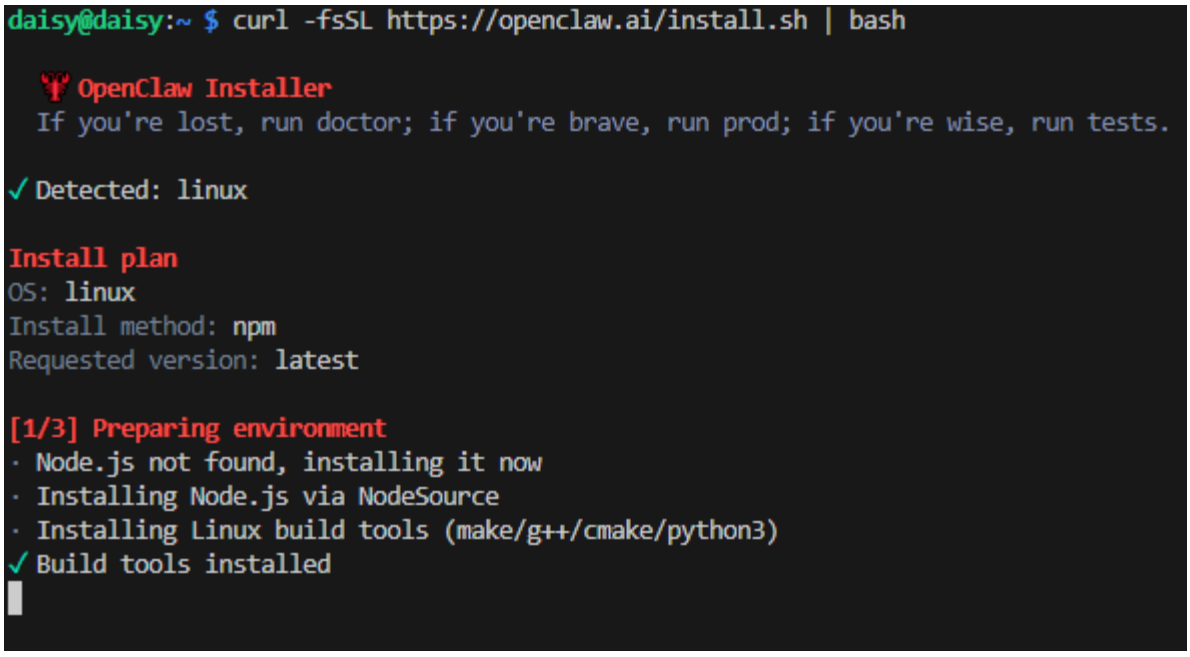
1. Open the terminal on your Raspberry Pi and run the following command directly. This command downloads the installation script from the official website and executes it:

```
curl -fsSL https://openclaw.ai/install.sh | bash
```

Note: Because new versions are updated rapidly, it's normal if your installation steps differ slightly.

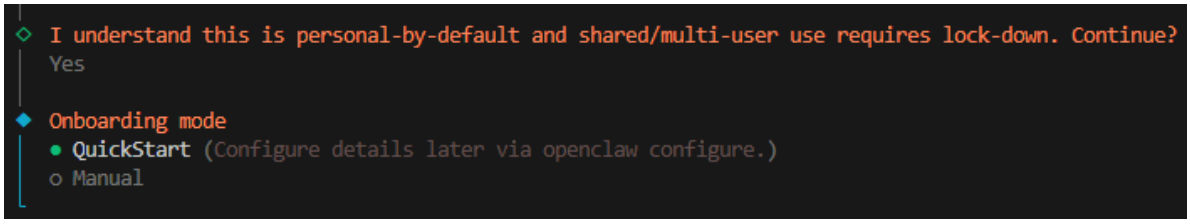
---

2. The script will automatically download and install OpenClaw.

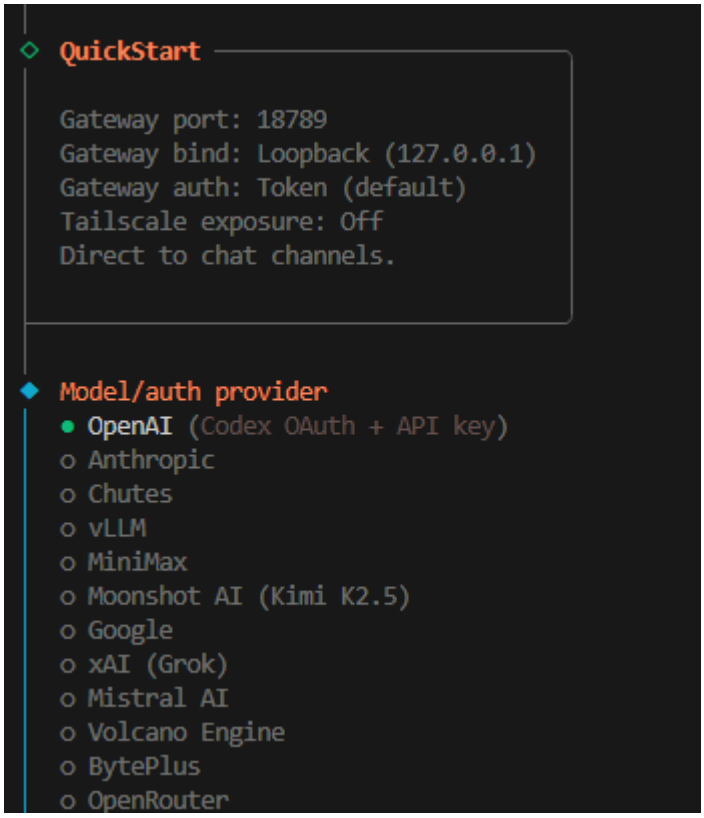


3. You will then see a security prompt asking if you trust OpenClaw. Once you are sure it is safe and reliable, use the arrow keys to navigate to "Yes" and press Enter.

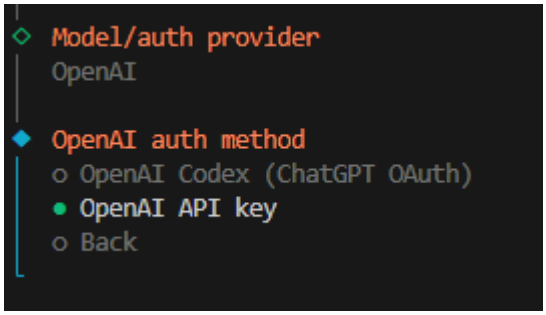
4. Select Quick Start, and then press Enter.



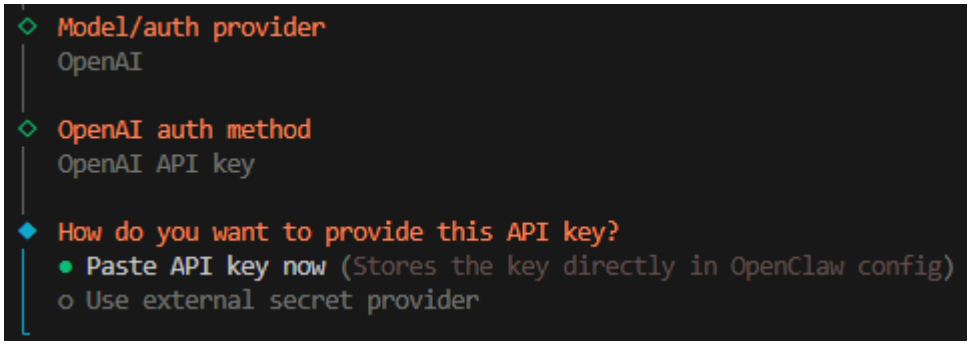
5. Select your Model, and then press Enter. Here we use OpenAI as an example.



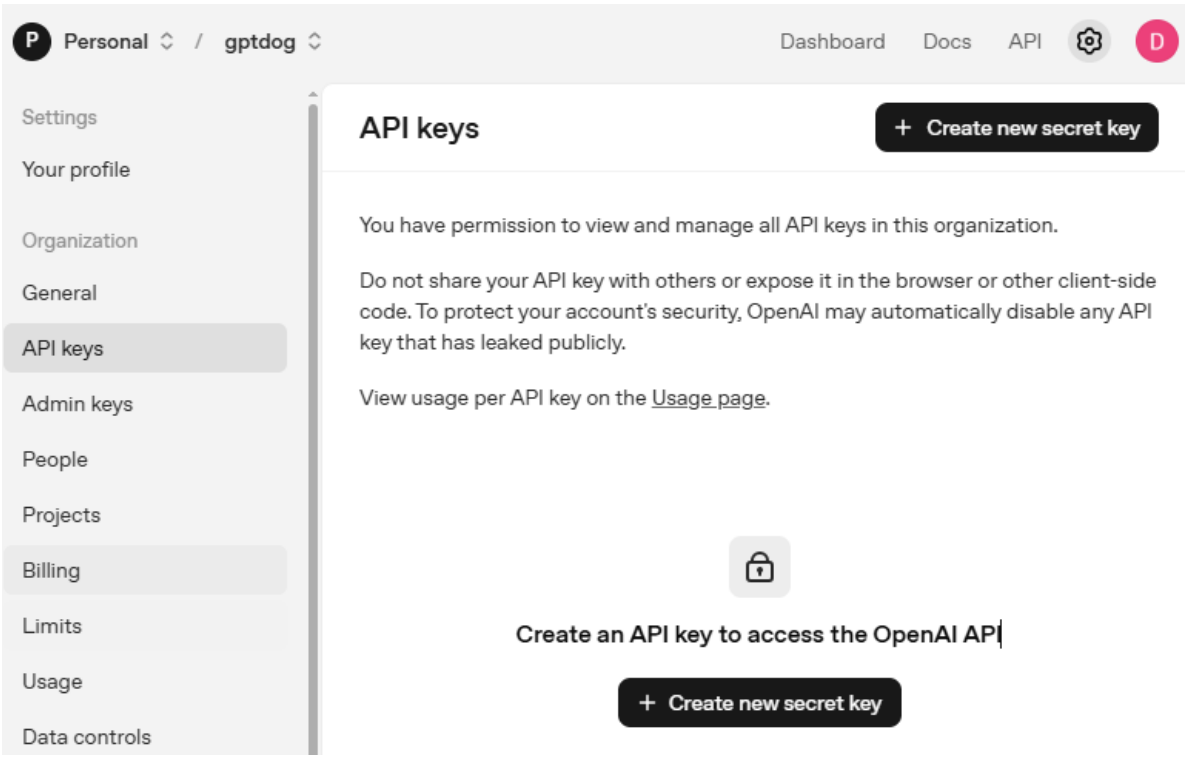
6. Select OpenAI API Key.



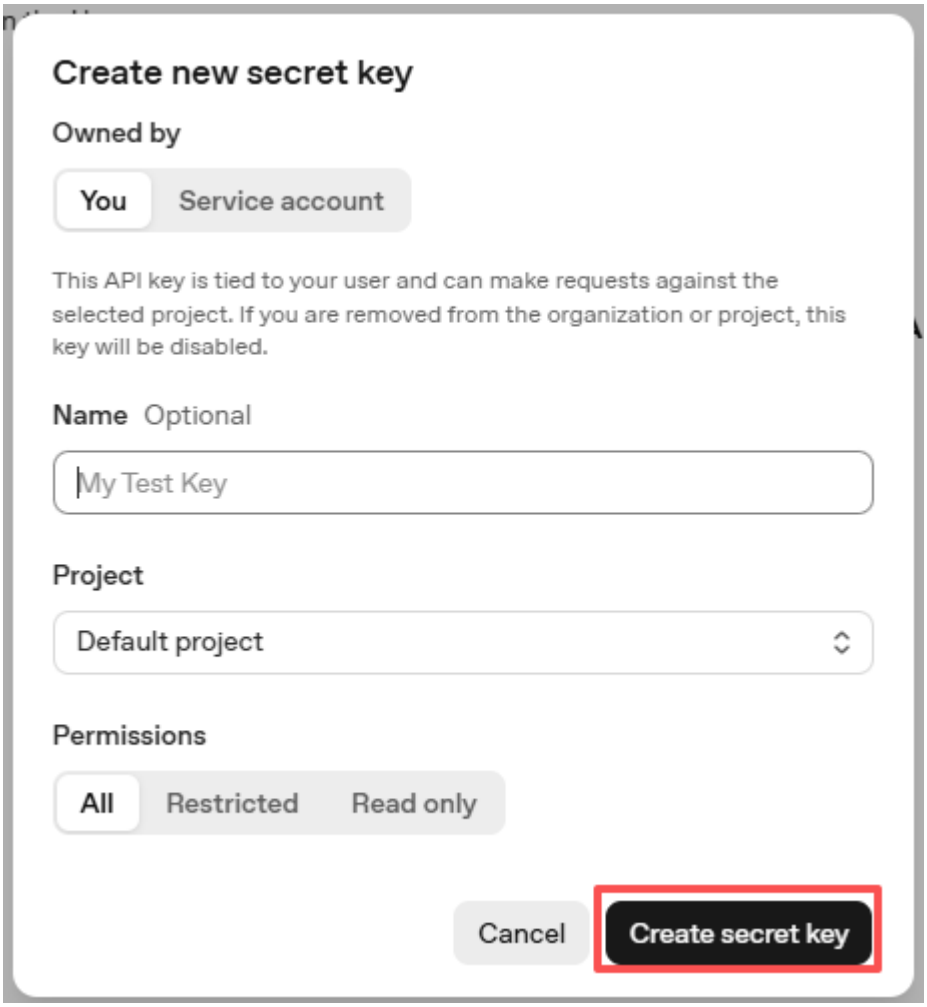
7. Paste API key now.



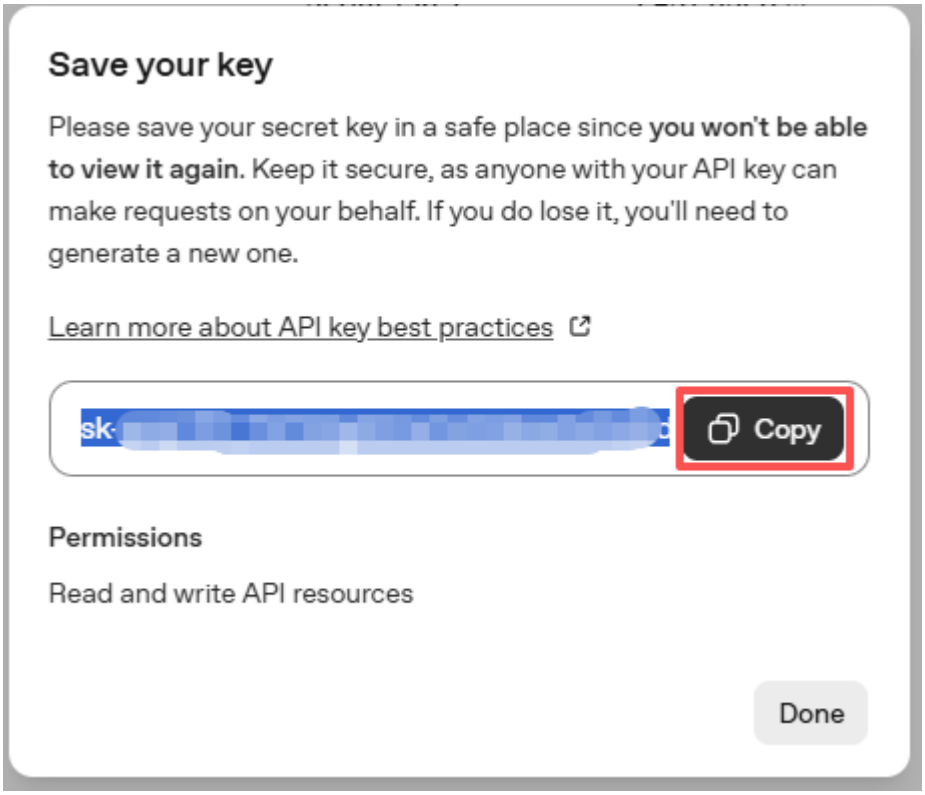
8. Go to and log in. On the **API keys** page, click **Create new secret key**.



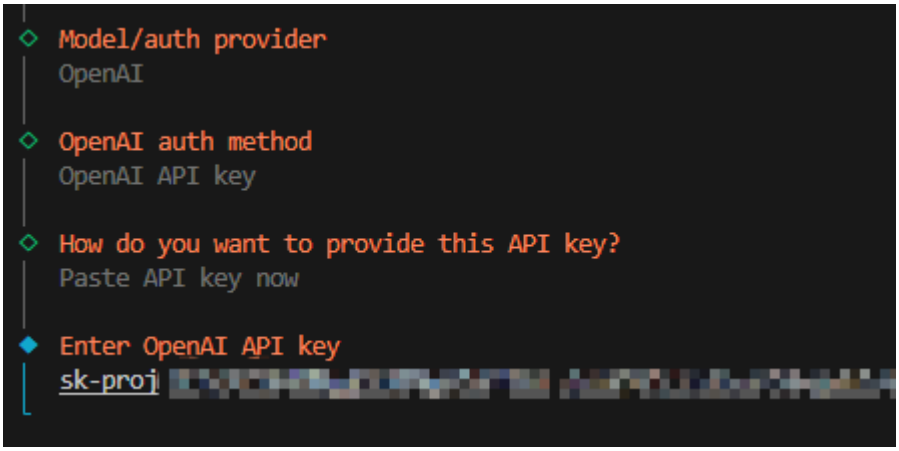
9. Fill in the details (Owner, Name, Project, and permissions if needed), then click **Create secret key**.



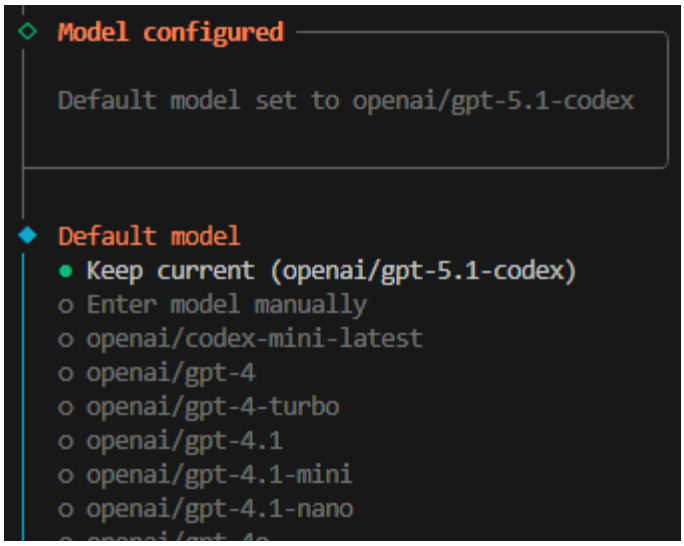
10. Once the key is created, copy it right away — you won't be able to see it again. If you lose it, you'll need to generate a new one.



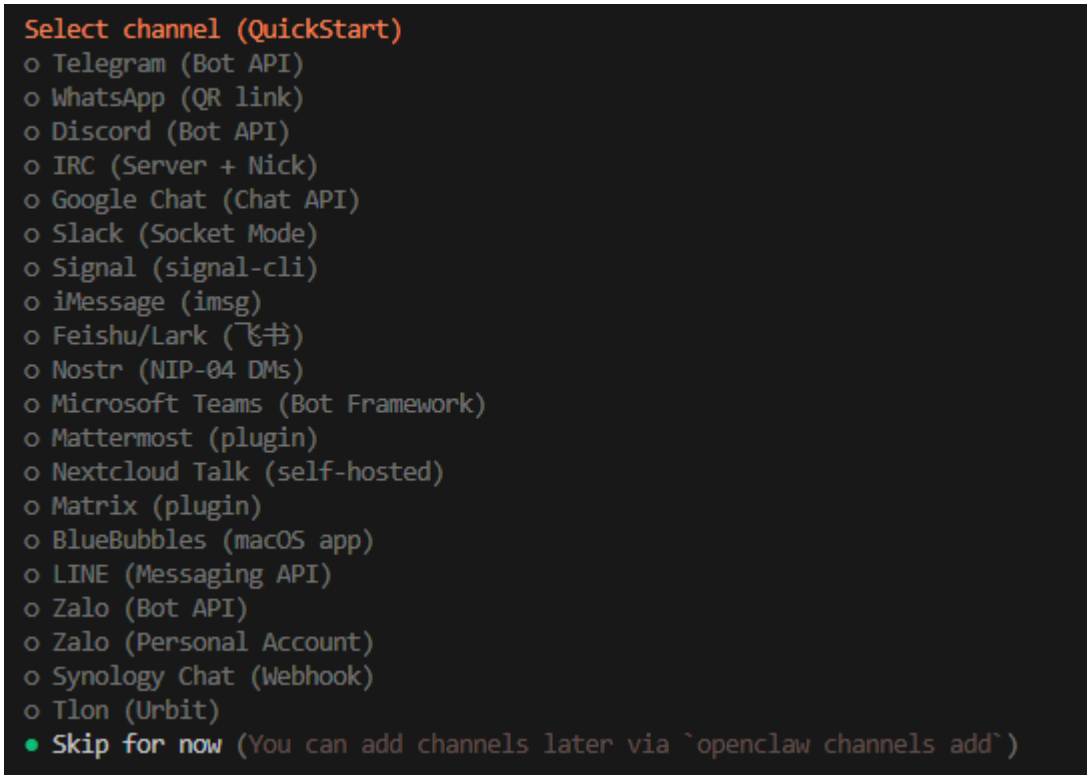
11. Paste the key into the OpenCLaw configuration.



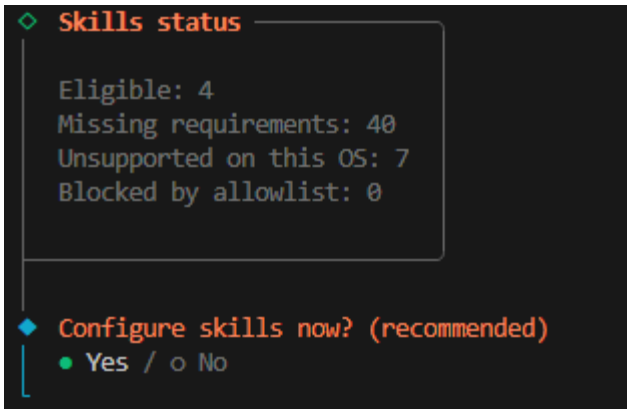
12. Select the Model you want to use. In this example, we will use **Keep current**.



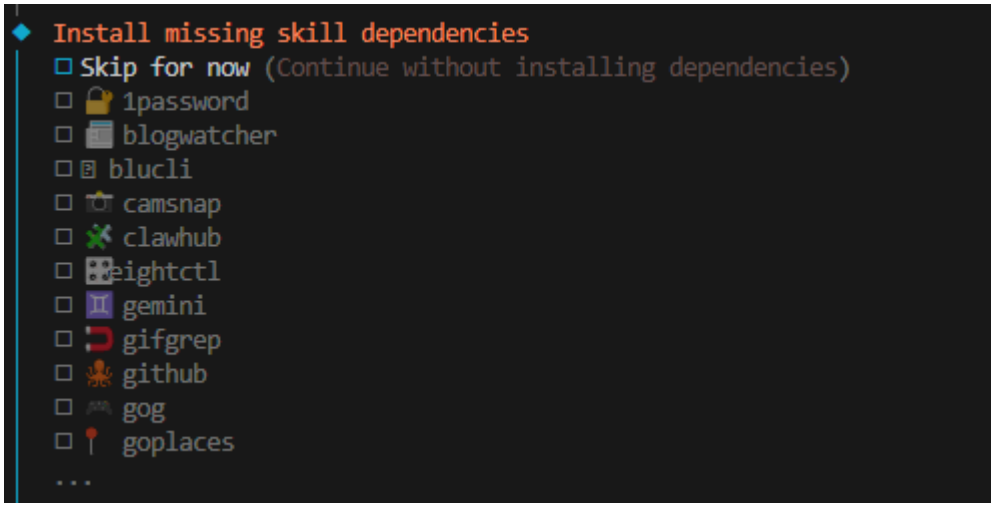
13. Next is the channel selection. Channels refer to the communication services supported by OpenClaw, such as Telegram, WhatsApp, Discord, and more. Use the down arrow key to select the "Skip for now" option, then press Enter.



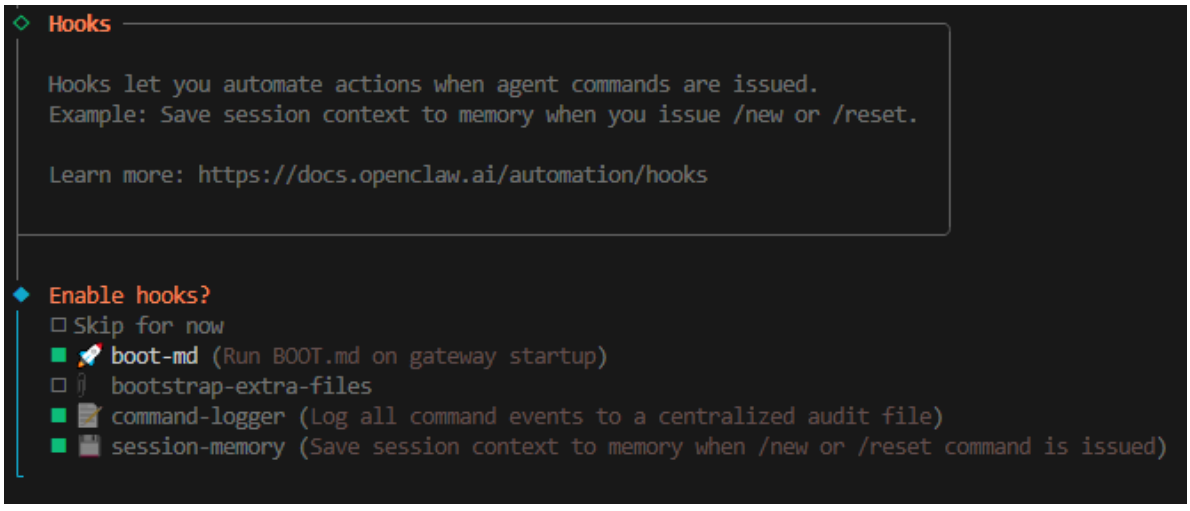
14. Next, you will be prompted to configure skills immediately. Select "Yes" and press Enter.



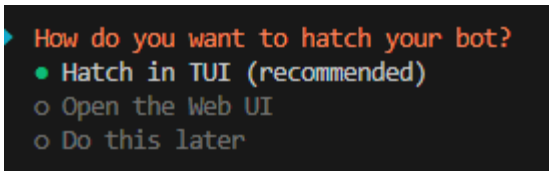
15. Install the skills you need. In the following example, we select the "Skip for now" option (press the spacebar to select), then press Enter.



16. Next are Hooks; we will check "command-logger" and "session-memory".



17. The installation is now complete. You can start OpenClaw by Selecting "Hatch in TUI" and pressing Enter.



Note: You can start OpenClaw by entering the following command:

```
openclaw tui
```

And You can press ctrl+c twice to exit the tui interface.

3.8.2 Making OpenClaw Operate the PiDog

What is PiDog Skill?

PiDog Skill is an extension for OpenClaw that allows you to control your SunFounder PiDog V2 robot dog through natural language. Instead of remembering complex command-line parameters, you can simply tell OpenClaw what you want PiDog to do — like "make the dog sit" or "turn the LED lights purple" — and OpenClaw will execute the appropriate commands automatically.

Here are some things you can do with PiDog Skill:

- **Basic Actions:** Make PiDog stand, sit, lie down, wag its tail, bark, walk forward/backward, or turn left/right
- **Pose Holding:** Keep PiDog in a specific pose (like standing) for extended periods
- **LED Light Control:** Change the eye colors with effects like breath, listen, boom, or solid light
- **Color Customization:** Choose from red, green, blue, yellow, purple, pink, cyan, white, orange, or custom hex colors

3.8.3 Prerequisites

Before you can use PiDog Skill with OpenClaw, make sure you have:

1. **PiDog V2** properly assembled and connected to your Raspberry Pi
2. **OpenClaw** installed and running
3. The following directories exist on your system:
   - `~/pidog`
   - `~/robot-hat`
   - `~/vilib`

You can verify the installation by running:

```
python3 -c "import pidog"
```

If this command runs without errors, you're ready to proceed.

3.8.4 Installing PiDog Skill

Follow these steps to install the PiDog Skill for OpenClaw:

1. **Create the skills directory** (if it doesn't already exist):

```
mkdir -p ~/.openclaw/workspace/skills/
```

2. **Copy the PiDog skill files** to the OpenClaw skills directory:

```
cp -r ~/pidog/pidog-control ~/.openclaw/workspace/skills/pidog-control/
```

Note: Replace `~/pidog-skill` with the actual path where your PiDog skill files are located.

3. **Verify the installation** by checking the skill files:

```
ls ~/.openclaw/workspace/skills/pidog-control/scripts/
```

You should see `pidog_ctl.py` and `pidog_rgb_ctl.py` in the output.

3.8.5 Testing PiDog Skill

Before using the skill with OpenClaw, it's recommended to test the basic functionality directly from the terminal.

Step 1: Check PiDog Status

```
python3 ~/.openclaw/workspace/skills/pidog-control/scripts/pidog_ctl.py status
```

Step 2: Run a Safe Test

```
python3 ~/.openclaw/workspace/skills/pidog-control/scripts/pidog_ctl.py safe-test
```

Step 3: Test Basic Actions

Make PiDog sit:

```
python3 ~/.openclaw/workspace/skills/pidog-control/scripts/pidog_ctl.py action sit
```

Make PiDog stand and hold the pose:

```
python3 ~/.openclaw/workspace/skills/pidog-control/scripts/pidog_ctl.py action stand --
↪hold
```

Make PiDog bark:

```
python3 ~/.openclaw/workspace/skills/pidog-control/scripts/pidog_ctl.py action bark
```

Step 4: Test LED Lights

Test the boom light effect with purple color:

```
python3 ~/.openclaw/workspace/skills/pidog-control/scripts/pidog_rgb_ctl.py light boom --
↪color purple
```

Test other light effects:

```
# Breath effect with red color
python3 ~/.openclaw/workspace/skills/pidog-control/scripts/pidog_rgb_ctl.py light breath␣
↪--color red

# Listen effect with blue color
python3 ~/.openclaw/workspace/skills/pidog-control/scripts/pidog_rgb_ctl.py light listen␣
↪--color blue

# Turn off lights
python3 ~/.openclaw/workspace/skills/pidog-control/scripts/pidog_rgb_ctl.py light off
```

3.8.6 Using PiDog Skill in OpenClaw

Once you've verified that PiDog Skill works from the command line, you can start using it within OpenClaw.

1. **Launch OpenClaw TUI**:

   ```
   openclaw tui
   ```

2. **Send natural language commands** to control PiDog. Here are some examples:
   - "Make the dog sit"
   - "Make PiDog stand and stay"
   - "Wag the dog's tail"
   - "Make the dog bark"
   - "Turn the LED lights purple with boom effect"
   - "Set the eye lights to breath effect with red color"
   - "Make PiDog walk forward"
3. **OpenClaw will automatically** translate your request into the appropriate command and execute it on PiDog.

3.8.7 Available Actions and Commands

Here is the complete list of supported actions for PiDog Skill:

Action	Description
stand	Make PiDog stand up
sit	Make PiDog sit down
lie	Make PiDog lie down
wag-tail	Wag PiDog's tail
bark	Make barking sound
forward	Walk forward
backward	Walk backward

Pose Holding:

Add `--hold` to any action to keep PiDog in that pose. For example: "stand –hold"

Light Effects:

Effect	Description
off	Turn off all LED lights
breath	Gentle breathing/pulsing effect
listen	Reactive listening mode
boom	Dynamic burst effect (most noticeable)
solid	Constant steady light (use boom for better effect)

Supported Colors:

red, green, blue, yellow, purple, pink, cyan, white, orange, or hex codes like `#FF5733`

3.8.8 Troubleshooting

OpenClaw Issues

Q. During installation, I get the error `Error: systemctl is-enabled unavailable: Command failed: systemctl --user is-enabled openclaw-gateway.service`. What should I do?

You can ignore this for now, but you might encounter issues in the next steps. Please refer to them one by one at that time.

Q. When I run `openclaw tui`, I get the error `-bash: openclaw: command not found`. What should I do?

Execute the following command:

```
echo 'export PATH="$HOME/.npm-global/bin:$PATH"' >> ~/.bashrc
source ~/.bashrc
```

You should now be able to start the tui interface with `openclaw tui`.

Q. In `openclaw tui`, I encounter `not connected to gateway - message not sent` or the message `gateway disconnected: closed`.

This is because your OpenClaw Gateway service is not started. Open another terminal and execute the following command to start the OpenClaw Gateway:

```
openclaw gateway
```

Then restart `openclaw tui`, and you can use it directly.

Q. I want to set the OpenClaw Gateway service to run in the background / start automatically on boot. How do I do that?

Normally, your OpenClaw Gateway service should start automatically on boot. If it doesn't, you can manually start it with the following command.

1. Create the `~/.config/systemd/user` directory:

```
mkdir -p ~/.config/systemd/user
```

2. Create the `openclaw-gateway.service` file:

```
cat > ~/.config/systemd/user/openclaw-gateway.service << EOF
[Unit]
Description=OpenClaw Gateway
After=network.target

[Service]
Type=simple
ExecStart=$HOME/.npm-global/bin/openclaw gateway run
Restart=on-failure
RestartSec=10
Environment="PATH=$HOME/.npm-global/bin:/usr/local/bin:/usr/bin:/bin:/usr/sbin"
Environment="NODE_ENV=production"

[Install]
WantedBy=default.target
EOF
```

3. Then reload the systemd configuration:

```
systemctl --user daemon-reload
```

4. Start the service:

```
systemctl --user start openclaw-gateway
```

At this point, restart `openclaw tui`, and you can use it directly.

5. Enable it to start on boot:

```
systemctl --user enable openclaw-gateway
```

Q. My OpenClaw can not operate the system, what should I do?

A newly installed OpenClaw may not have permission to operate your Raspberry Pi system by default; it can only chat. We need to manually configure the permissions.

1. Open the OpenClaw configuration file:

```
nano ~/.openclaw/openclaw.json
```

2. Find the `tools` option and change the `profile` and `exec` as shown.

```
"tools": {
    "profile": "coding",
    "exec": {
        "secrity": "full"
    }
},
```

3. Save and exit.

4. Enter the following command in the terminal to restart the OpenClaw Gateway:

```
openclaw gateway restart
```

Now, OpenClaw should have read and write permissions and be able to operate your Raspberry Pi system.

PiDog Issues

Q. PiDog doesn't respond to commands. What should I do?

First, verify that PiDog is properly connected and powered on. Then test the basic command:

```
python3 ~/.openclaw/workspace/skills/pidog-control/scripts/pidog_ctl.py status
```

If this fails, check that the required directories exist:

- `~/pidog`
- `~/robot-hat`
- `~/vilib`

Q. The `import pidog` test fails.

This means the PiDog Python library is not properly installed. Please refer to the PiDog V2 official installation guide to install the necessary libraries.

Q. LED lights don't work as expected.

If solid light doesn't show clearly, use the `boom` effect instead — it produces the most noticeable results:

```
python3 ~/.openclaw/workspace/skills/pidog-control/scripts/pidog_rgb_ctl.py light␣
↪boom --color purple
```

Q. OpenClaw doesn't recognize the PiDog skill.

Remind OpenClaw to sync the skills by saying in the TUI: *"Please rsync my skills"* or restart OpenClaw gateway:

```
openclaw gateway restart
```

Q. The bark action doesn't sound right.

The bark action uses the `single_bark_1` sound by default. This is normal behavior for PiDog V2.

CHAPTER

FOUR

HARDWARE

When you are writing code, you may need to know how each module works or the role of each pin, then please see this chapter.

In this chapter you will find a description of each module's function, technical parameters and working principle.

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

4.1 Robot HAT

For V2 Version:

is a multifunctional expansion board that allows Raspberry Pi to be quickly turned into a robot. The V2 Robot HAT integrates a microphone into the Standard version's architecture, enabling audio recording and voice recognition capabilities.

For detailed instructions, please refer to: .

For Standard Version:

is a multifunctional expansion board that allows Raspberry Pi to be quickly turned into a robot. An MCU is on board to extend the PWM output and ADC input for the Raspberry Pi, as well as a motor driver chip, I2S audio module and mono speaker. As well as the GPIOs that lead out of the Raspberry Pi itself.

It also comes with a Speaker, which can be used to play background music, sound effects and implement TTS functions to make your project more interesting.

Accepts 7-12V power input with 2 battery indicators, 1 charge indicator and 1 power indicator. The board also has a user available LED and a button for you to quickly test some effects.

For detailed instructions, please refer to: .

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

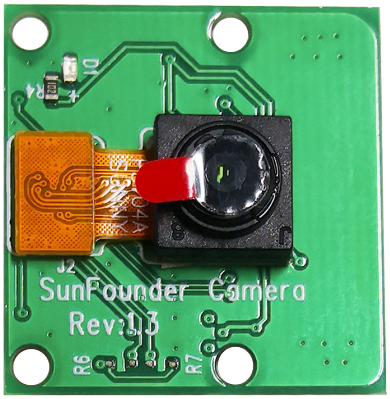
Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

4.2 Camera Module

Description



This is a 5MP Raspberry Pi camera module with OV5647 sensor. It's plug and play, connect the included ribbon cable to the CSI (Camera Serial Interface) port on your Raspberry Pi and you're ready to go.

The board is small, about 25mm x 23mm x 9mm, and weighs 3g, making it ideal for mobile or other size and weight-critical applications. The camera module has a native resolution of 5 megapixels and has an on-board fixed focus lens that captures still images at 2592 x 1944 pixels, and also supports 1080p30, 720p60 and 640x480p90 video.

Note: The module is only capable of capturing pictures and videos, not sound.

Specification

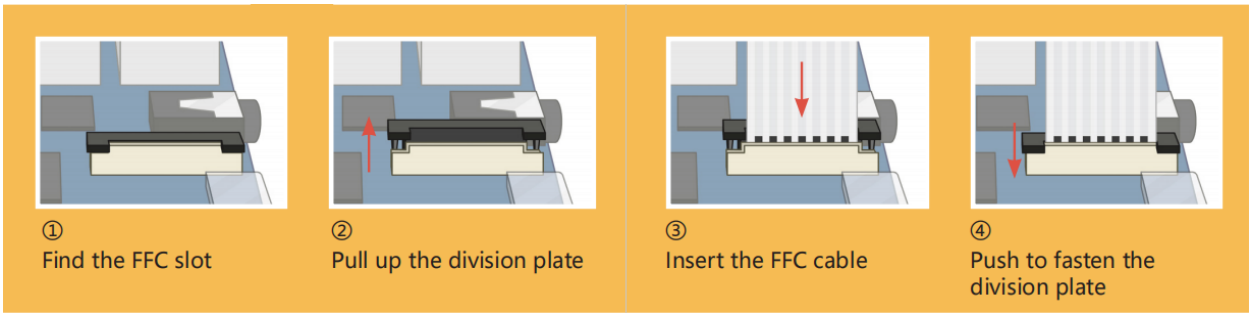
- **Static Images Resolution**: 2592×1944
- **Supported Video Resolution**: 1080p/30 fps, 720p/ 60fps and 640 x480p 60/90 video recording
- **Aperture (F)**: 1.8
- **Visual Angle**: 65 degree
- **Dimension**: 24mmx23.5mmx8mm
- **Weight**: 3g

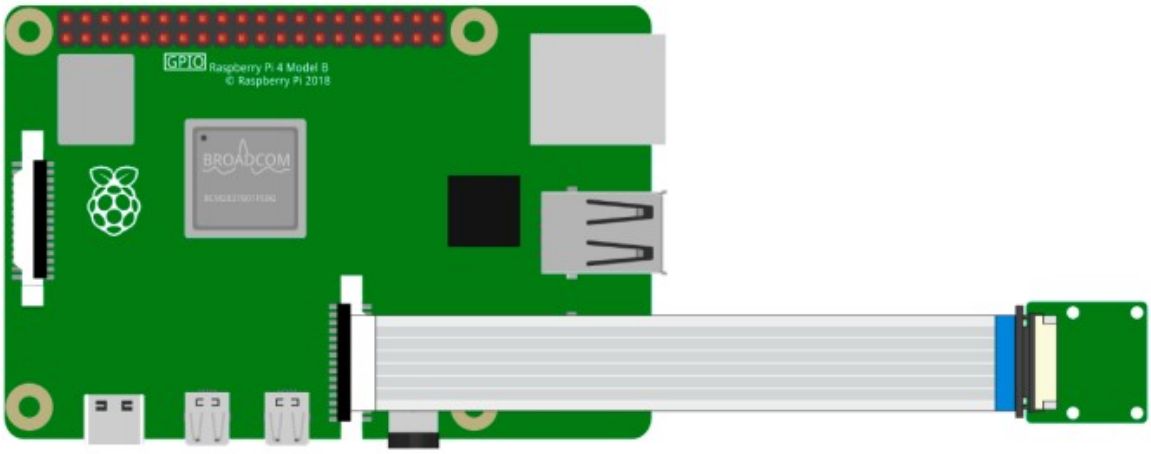
- **Interface**: CSI connector
- **Supported OS**: Raspberry Pi OS(latest version recommended)

Assemble the Camera Module

On the camera module or Raspberry Pi, you will find a flat plastic connector. Carefully pull out the black fixing switch until the fixing switch is partially pulled out. Insert the FFC cable into the plastic connector in the direction shown and push the fixing switch back into place.

If the FFC wire is installed correctly, it will be straight and will not pull out when you gently pull on it. If not, reinstall it again.





Warning: Do not install the camera with the power on, it may damage your camera.

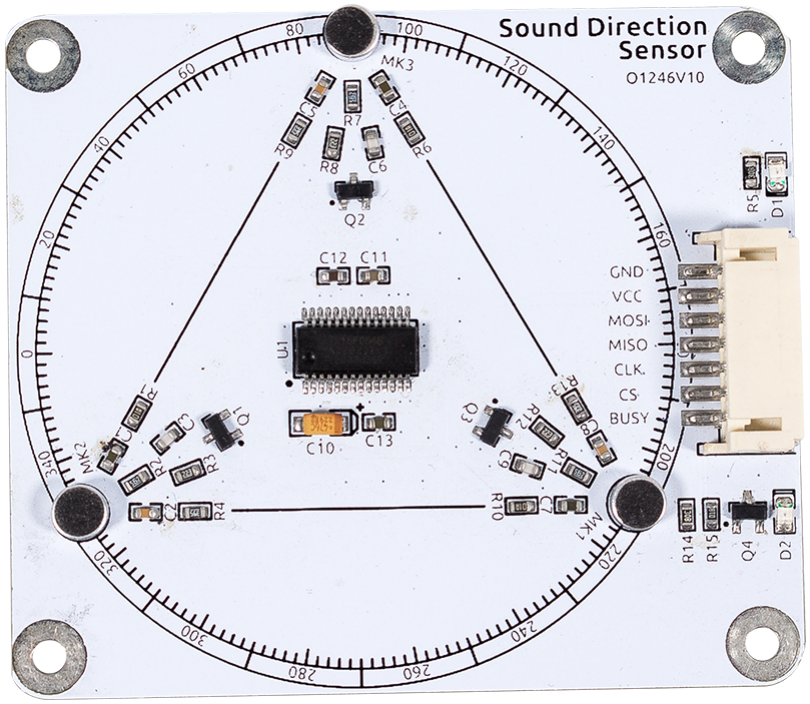
Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

4.3 Sound Direction Sensor



This is a sound direction recognition module. It is equipped with 3 microphones, which can detect sound sources from all directions, and is equipped with a TR16F064B, which is used to process sound signals and calculate the sound source direction. The minimum reconnaissance unit of this module is 20 degrees, and the data range is 0~360.

Data transmission process: the main controller pulls up the BUSY pin, and TR16F064B starts to monitor the direction. When 064B recognizes the direction, it will pull down the BUSY pin; When the main control detects that BUSY is low, it will send 16bit arbitrary data to 064B (follow the MSB transmission), and accept 16bit data, which is the sound direction data processed by 064B. After completion, the main control will pull the BUSY pin high to detect the direction again.

Specifications

- Power supply: 3.3V
- Communication: SPI
- Connector: PH2.0 7P
- Sound recognition angle range 360°
- Voice recognition angular accuracy ~10°

Pin Out

- GND - Ground Input
- VCC - 3.3V Power Supply Input
- MOSI - SPI MOSI

- MISO - SPI MISO
- SCLK - SPI clock
- CS - SPI Chip Select
- BUSY - busy detection

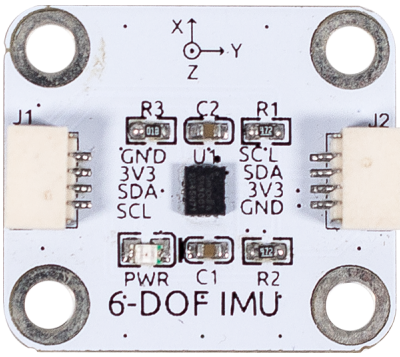
Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

4.4 6-DOF IMU



The 6-DOF IMU is based on the SH3001.

SH3001 is a six-axis IMU (Inertial measurement unit). It integrates a three-axis gyroscope and a three-axis accelerometer. It is small in size and low in power consumption. It is suitable for consumer electronics market applications and can provide high-precision real-time angular velocity and linear acceleration data. The SH3001 has excellent temperature stability and can maintain high resolution within the operating range of -40°C to 85°C.

It is typically used in smartphones, tablet computers, multi-rotor drones, smart sweepers, page-turning laser pointers, AR/VR, smart remote controls, smart bracelets and other products.

Specifications

- Power Supply: 3.3V
- Communication: IIC
- Connector: SH1.0 4P

Pin Out

- GND - Ground Input

- VCC - Power Supply Input
- SDA - IIC SDA
- SCL - IIC SCL

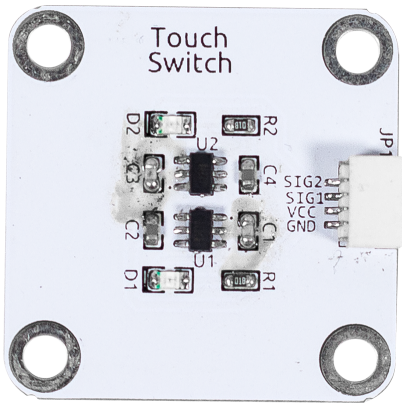
Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

4.5 Dual Touch Sensor



Dual channel touch sensor, based on two ttp223 touch sensors. When a touch signal is detected, the corresponding pin level will be pulled low.

TTP223 is a touch pad detector IC that provides 1 touch key. The touch detection IC is specially designed to replace the traditional direct keys with different pad sizes. It features low power consumption and wide operating voltage.

Specifications

- Power Supply: 2.0V~5.5V
- Signal Output: Digital signal
- Connector: SH1.0 4P

Pin Out

- GND - Ground Input
- VCC - Power Supply Input
- SIG1 - Touch signal 1, low level means touch

- SIG2 - Touch signal 2, low level means touch

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

4.6 11-channel Light Board

This is an 11-channel RGB LED module, which is equipped with 11 RGB LEDs controlled by the SLED1735 chip.

SLED1734 can drive up to 256 LEDs and 75 RGB LEDs. In the LED matrix controlled by SLED1734, each LED has on/off, blinking, breathing light and automatic synchronization and many other functions. The chip has built-in PWM (pulse width modulation) technology, which can provide 256 levels of brightness adjustment. It also has a 16-level dot correction function.

Specifications

- Power supply: 3.3V
- Communication: IIC
- Connector: SH1.0 4P
- LEDs: 3535 RGB LEDs

Pin Out

- GND - Ground Input
- VCC - Power Supply Input
- SDA - IIC SDA
- SCL - IIC SCL

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

4.7 Ultrasonic Module

- **TRIG**: Trigger Pulse Input
- **ECHO**: Echo Pulse Output
- **GND**: Ground
- **VCC**: 5V Supply

This is the HC-SR04 ultrasonic distance sensor, providing non-contact measurement from 2 cm to 400 cm with a range accuracy of up to 3 mm. Included on the module is an ultrasonic transmitter, a receiver and a control circuit.

You only need to connect 4 pins: VCC (power), Trig (trigger), Echo (receive) and GND (ground) to make it easy to use for your measurement projects.

Features

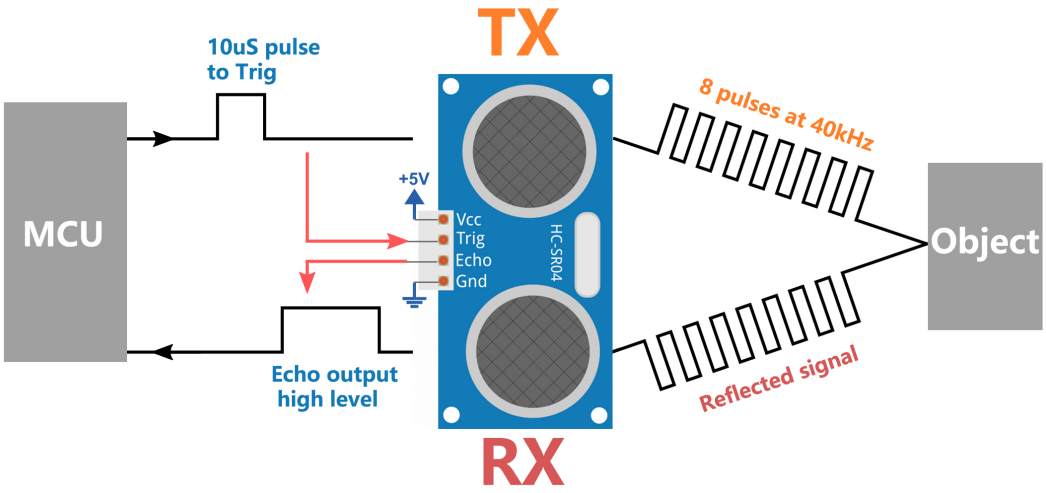
- Working Voltage: DC5V
- Working Current: 16mA
- Working Frequency: 40Hz
- Max Range: 500cm
- Min Range: 2cm

- Trigger Input Signal: 10uS TTL pulse
- Echo Output Signal: Input TTL lever signal and the range in proportion
- Connector: XH2.54-4P
- Dimension: 46x20.5x15 mm

Principle

The basic principles are as follows:

- Using IO trigger for at least 10us high level signal.
- The module sends an 8 cycle burst of ultrasound at 40 kHz and detects whether a pulse signal is received.
- Echo will output a high level if a signal is returned; the duration of the high level is the time from emission to return.
- Distance = (high level time x velocity of sound (340M/S)) / 2



Formula:

- us / 58 = centimeters distance
- us / 148 = inch distance
- distance = high level time x velocity (340M/S) / 2

Application Notes

- This module should not be connected under power up, if necessary, let the module's GND be connected first. Otherwise, it will affect the work of the module.
- The area of the object to be measured should be at least 0.5 square meters and as flat as possible. Otherwise, it will affect results.

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.

- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

4.8 3-pin Battery

- **VCC**: Battery positive terminal, here there are two sets of VCC and GND is to increase the current and reduce the resistance.
- **Middle**: To balance the voltage between the two cells and thus protect the battery.
- **GND**: Negative battery terminal.

This is a custom battery pack made by SunFounder consisting of two 18650 batteries with a capacity of 2000mAh. The connector is XH2.54 3P, which can be charged directly after being inserted into the Robot HAT.

Features

- Composition: Li-ion
- Battery Capacity: 2000mAh, 14.8Wh
- Battery Weight: 90.8g
- Number of Cells: 2
- Connector: XH2.54 3P
- Over-discharge protection: 6.0V

CHAPTER

FIVE

APPENDIX

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

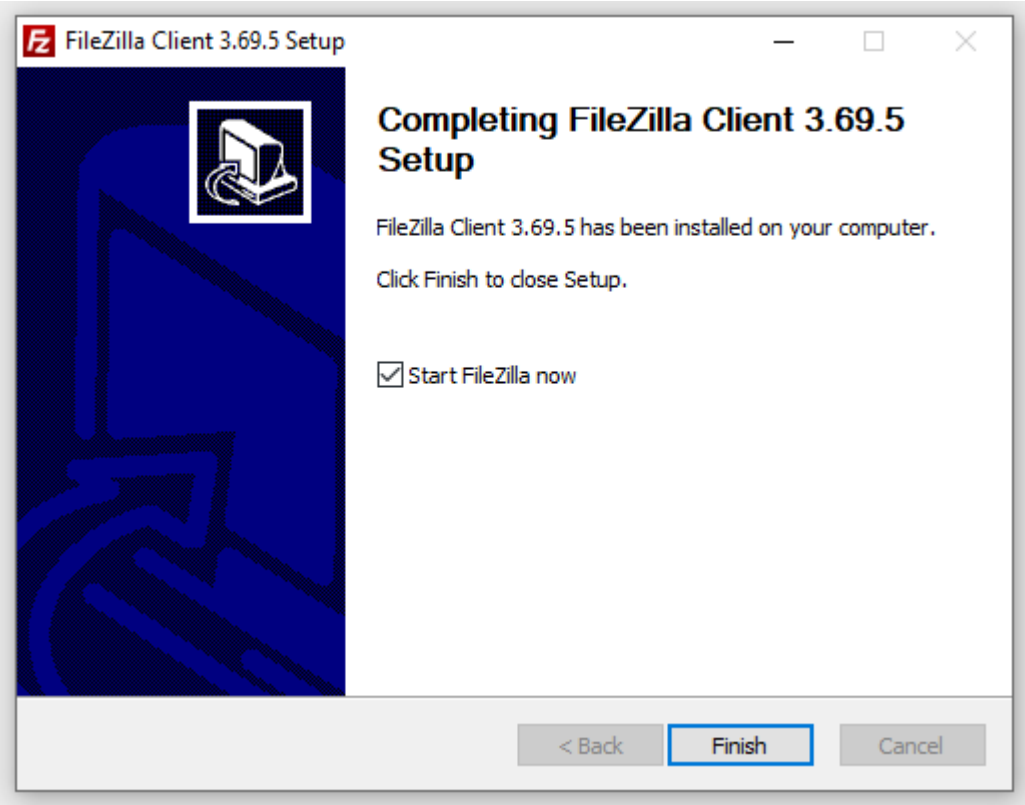
Ready to explore and create with us? Click [] and join today!

5.1 FileZilla Software

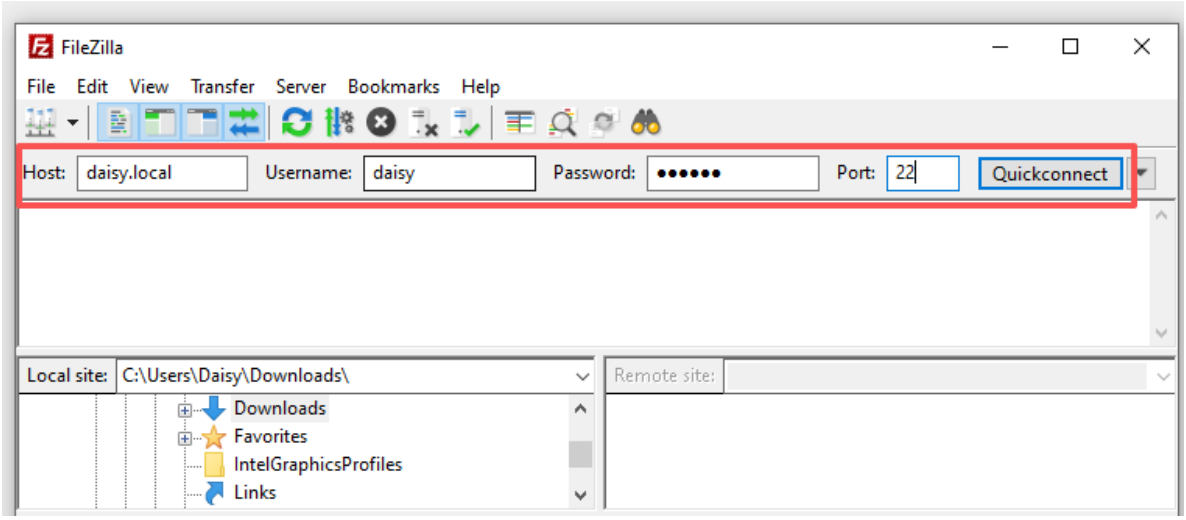
File Transfer Protocol (FTP) is commonly used to transfer files between computers over a network. **FileZilla** is an open-source client that supports FTP, FTPS, and **SFTP** (recommended for Raspberry Pi). With FileZilla, you can easily upload files (such as images, audio, and scripts) from your computer to the Raspberry Pi, or download files from the Pi back to your computer.

5.1.1 Download FileZilla

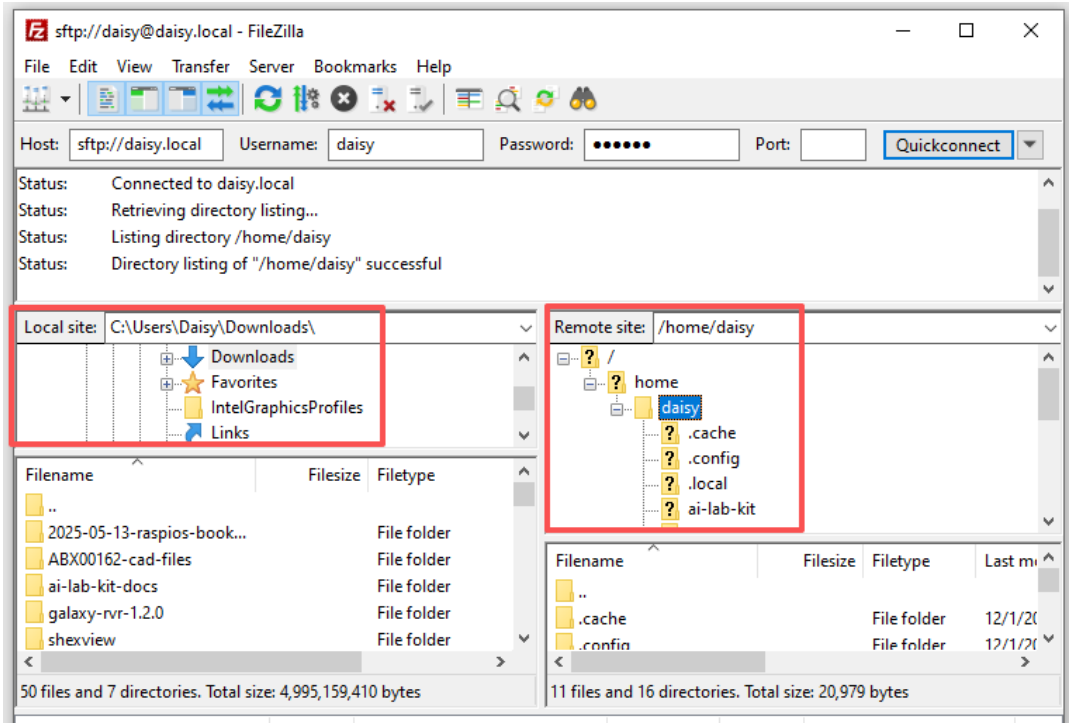
1. Visit the official website and download **FileZilla Client** for your operating system.
2. Install and launch the program.



3. Open FileZilla and enter the following information:
   - **Host:** `<hostname>.local` or the Raspberry Pi's IP address
   - **Username:** your Pi username
   - **Password:** the password set in Raspberry Pi Imager
   - **Port:** `22` (for SFTP)
   - Click **Quickconnect** (or press **Enter**) to establish a connection.



4. Once connected, the left panel shows your **local files**, and the right panel shows the **Raspberry Pi files**.



5. You can:

   - **Upload** a file: drag from the left panel → right panel
   - **Download** a file: drag from the right panel → left panel

   FileZilla will immediately start the transfer, and the status will appear in the panel at the bottom.

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.

- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

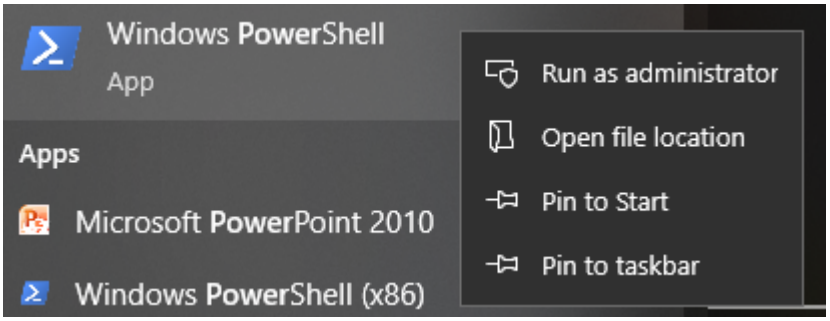
5.2 Install OpenSSH via PowerShell

If you see the following error when running `ssh <username>@<hostname>.local` or `ssh <username>@<IP>`:

```
ssh: The term 'ssh' is not recognized as the name of a cmdlet, function, script file, or␣
↪operable program.
```

It means your Windows system does not have OpenSSH installed. Follow the steps below to install it manually.

1. Open the Windows Start Menu, type **powershell**, right-click **Windows PowerShell**, and select **Run as administrator**.



2. Install the OpenSSH Client:

```
Add-WindowsCapability -Online -Name OpenSSH.Client~~~~0.0.1.0
```

3. After installation, you should see output similar to:

```
Path          :
Online        : True
RestartNeeded : False
```

4. Verify the installation:

```
Get-WindowsCapability -Online | Where-Object Name -like 'OpenSSH*'
```

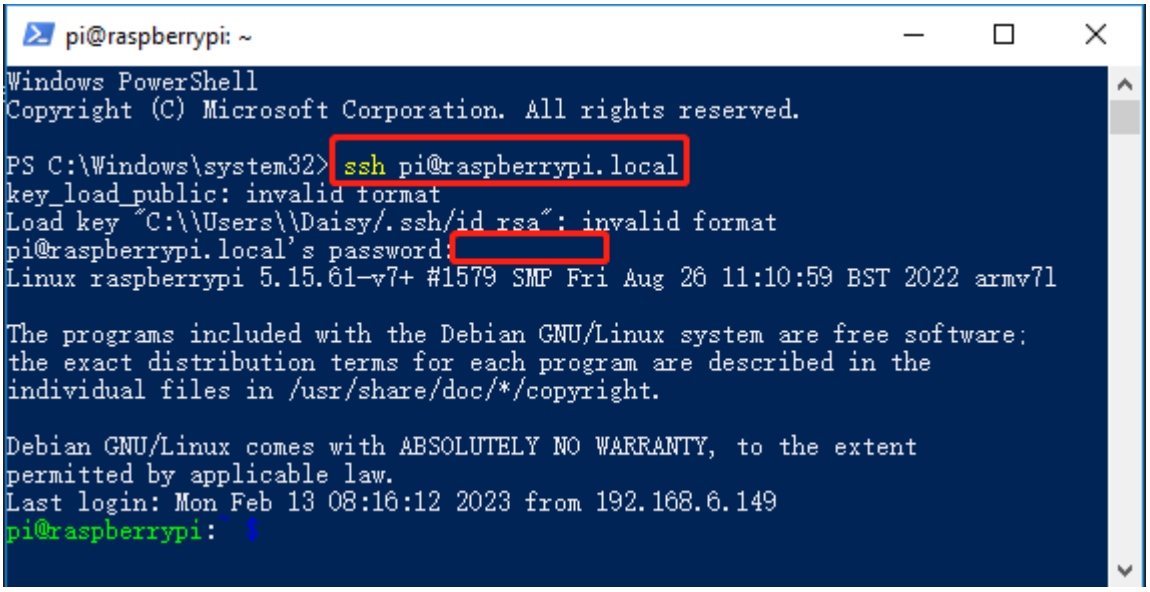
5. If OpenSSH is installed, the output will include:

```
Name  : OpenSSH.Client~~~~0.0.1.0
State : Installed
Name  : OpenSSH.Server~~~~0.0.1.0
State : NotPresent
```

> **Warning:** If `Installed` does not appear, your Windows system may be too old. In this case, we recommend using a third-party SSH tool. See: login_windows

6. Close PowerShell, reopen it (no need to run as administrator this time), and use the `ssh` command to log in:

```
ssh <username>@<hostname>.local
```



Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

5.3 Remote Desktop

You can access and control the Raspberry Pi desktop remotely from another computer. The recommended method is **VNC**, which is officially supported on Raspberry Pi OS and provides a reliable and consistent desktop experience.

The following section explains how to enable VNC on your Raspberry Pi and connect to it using .

5.3.1 Enable the VNC Service

RealVNC Server is preinstalled on Raspberry Pi OS, but it is **disabled by default**. You must enable it through the configuration tool.

1. Open a terminal on your computer (Windows: **PowerShell**; macOS/Linux: **Terminal**) and connect to your Raspberry Pi:

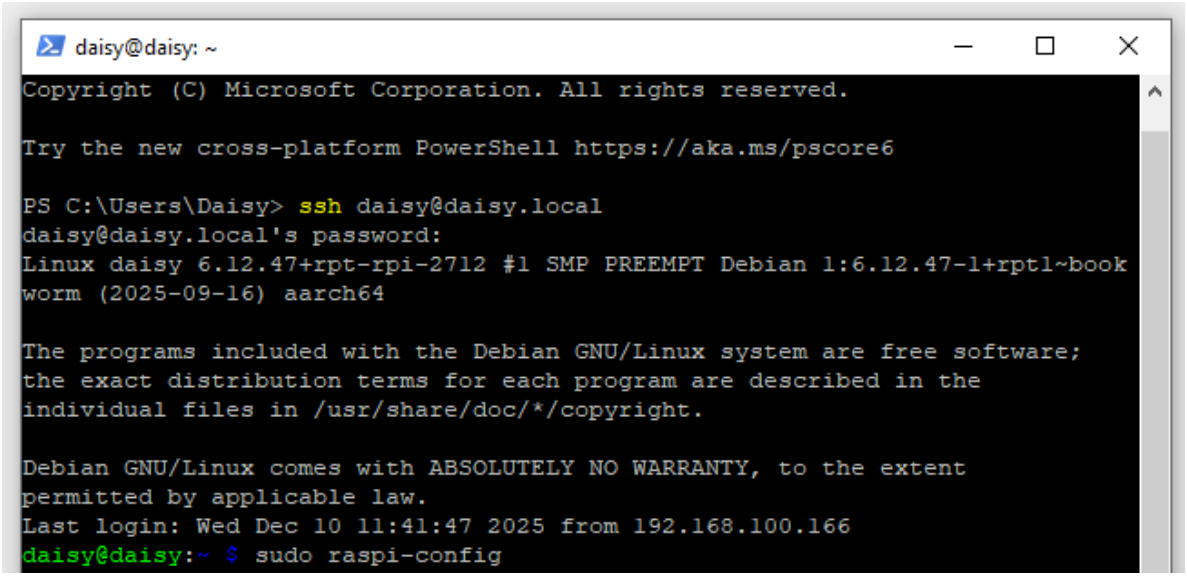
   ```
   ssh <username>@<hostname>.local
   ```

   or

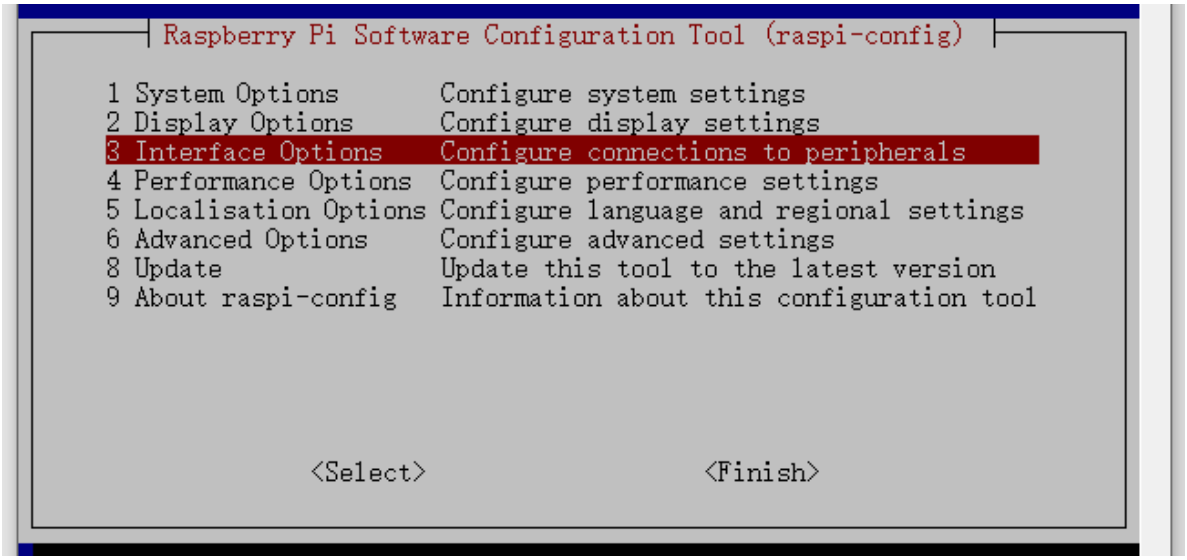
   ```
   ssh <username>@<ip_address>
   ```

2. Run the configuration tool:

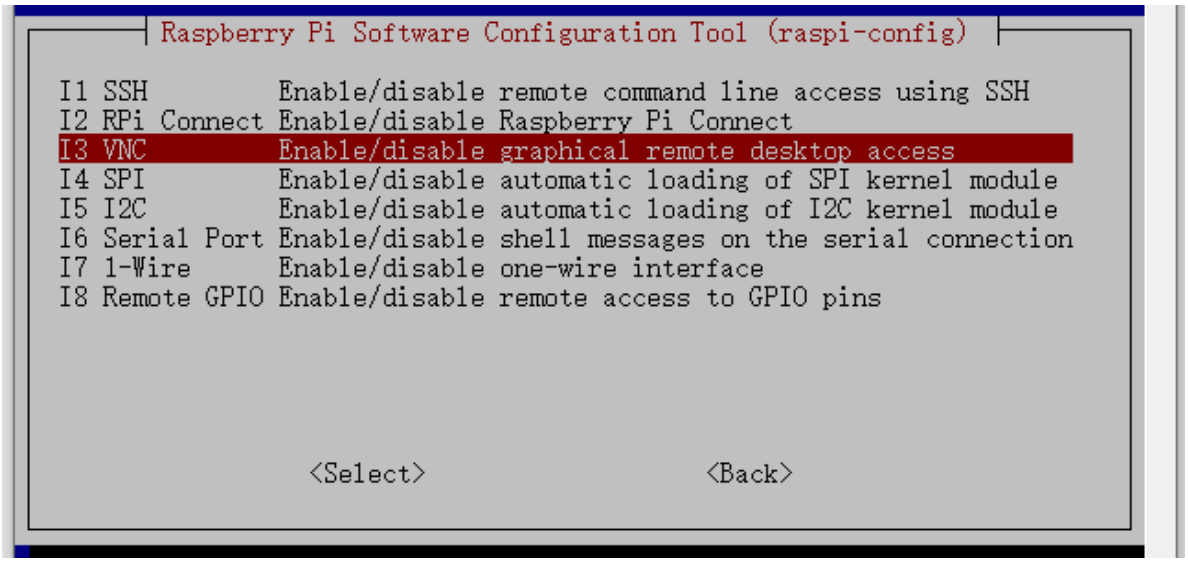
   ```
   sudo raspi-config
   ```



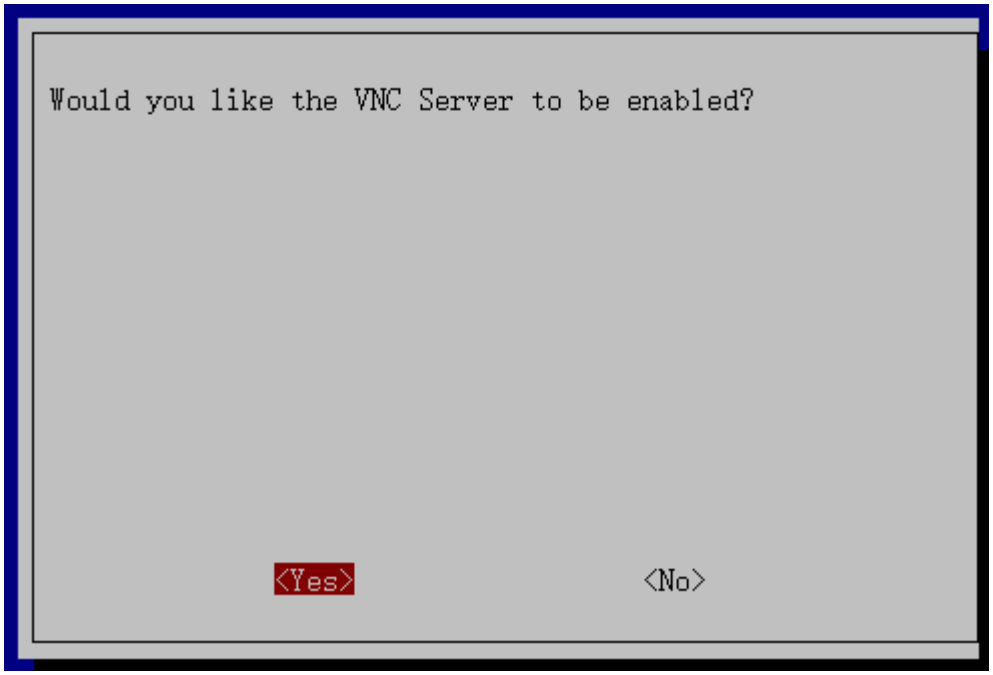
3. Select **Interfacing Options** and press **Enter**.



4. Select **VNC**.

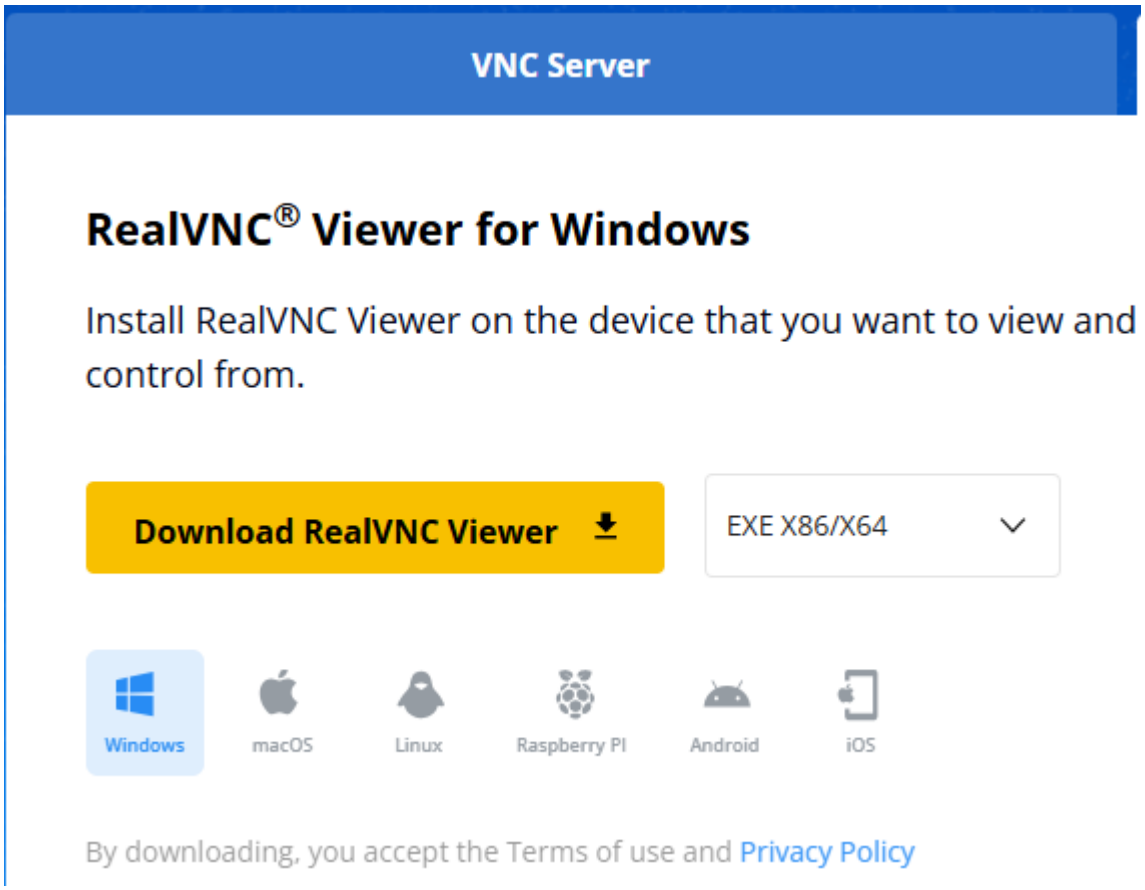


5. Choose **Yes**, then **OK**, and finally **Finish** to exit.

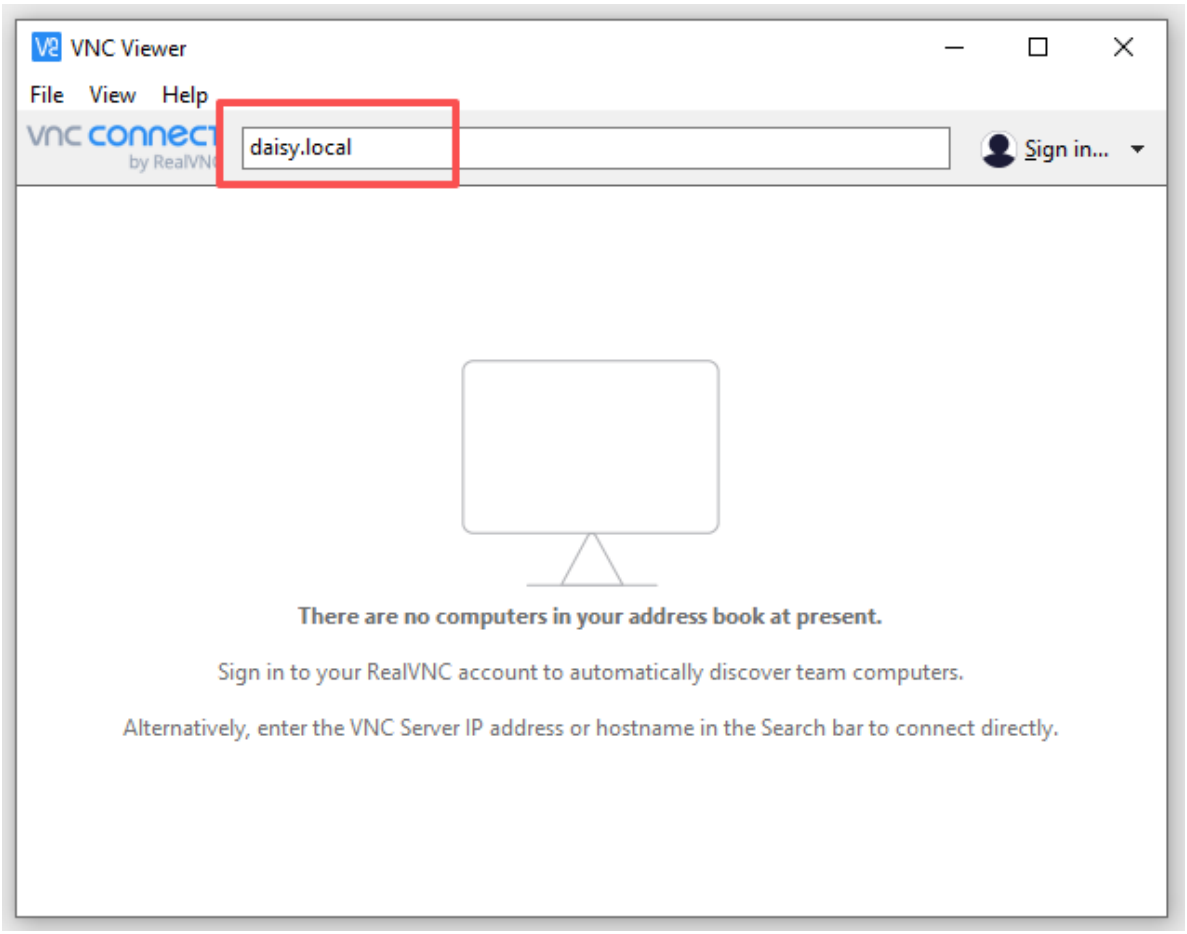


5.3.2 Log in with RealVNC® Viewer

1. Download and install for your operating system.

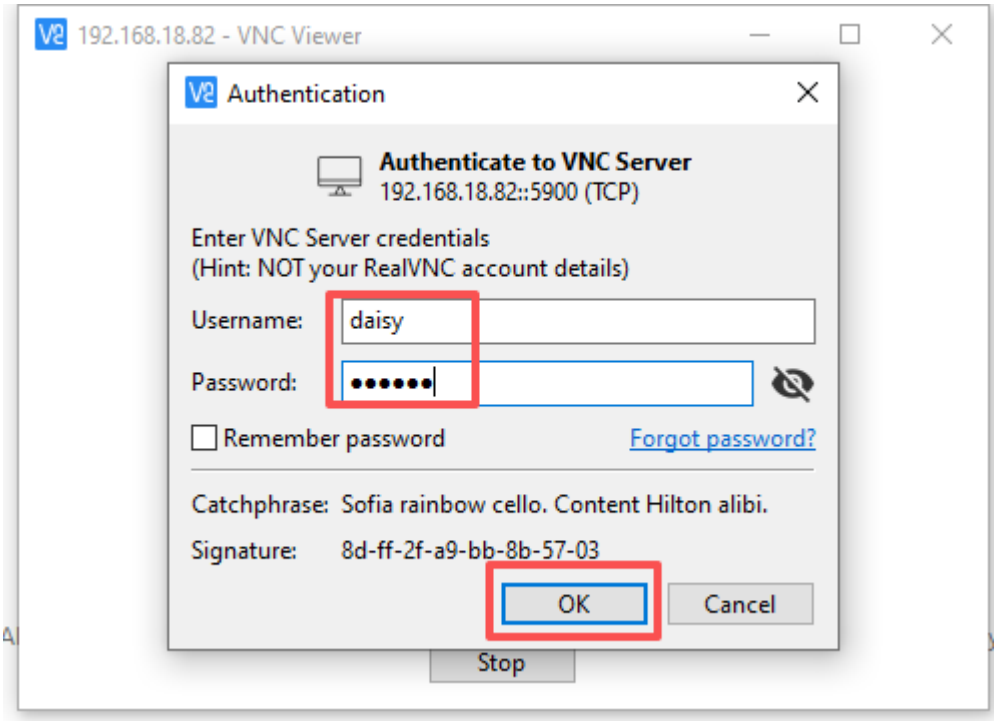


2. Open **RealVNC Viewer**, then enter your Raspberry Pi's IP address or `<hostname>.local` and press **Enter**.

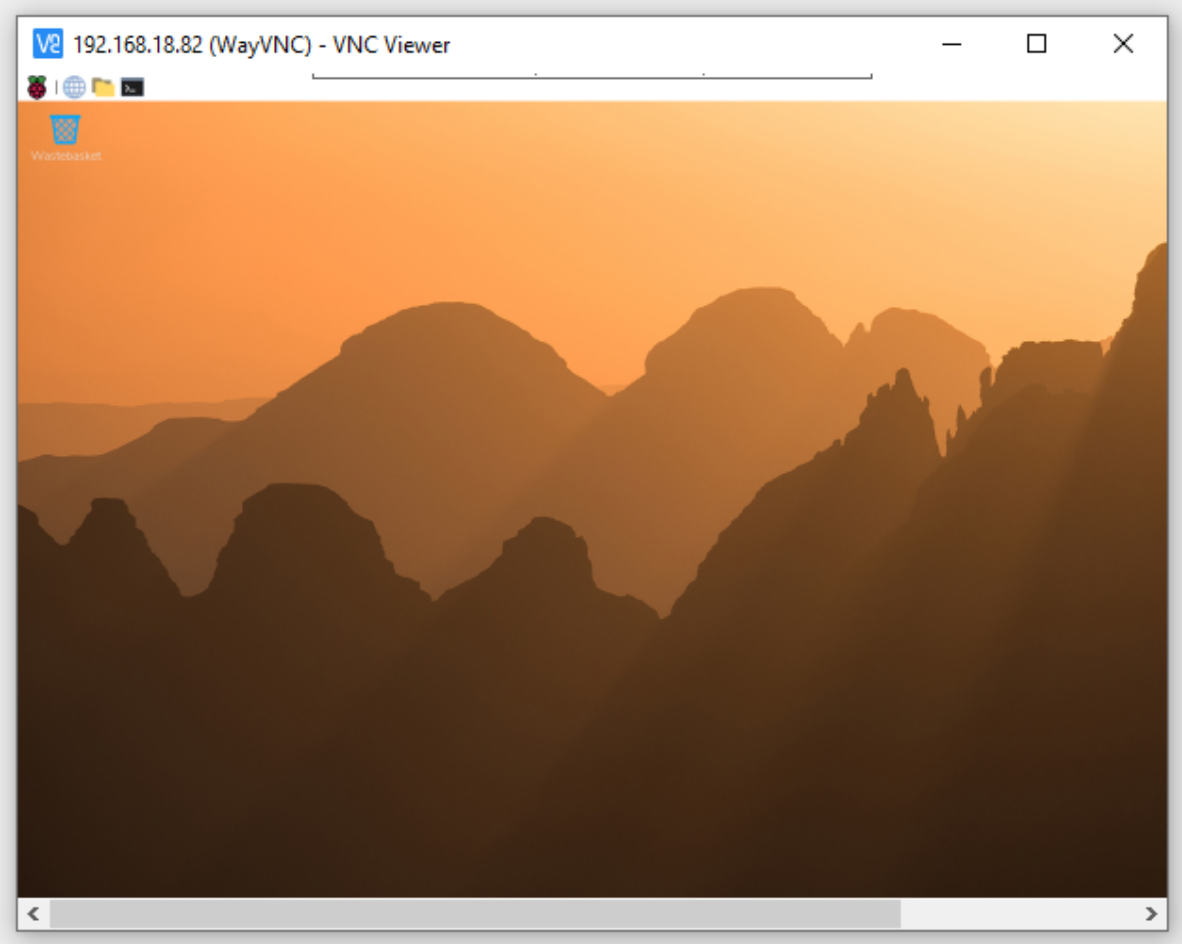


3. Enter your Raspberry Pi's **username** and **password**, then select **OK**.

Note: When connecting for the first time, you may see a message such as "VNC Server not recognized". Select **Continue** to proceed.



4. You should now see the Raspberry Pi desktop:



This completes the VNC setup process.

5.3.3 Additional Notes

- **Desktop version required**
  - VNC requires the Raspberry Pi to be running the full desktop version of Raspberry Pi OS.
  - If you are using **Raspberry Pi OS Lite**, install VNC Server manually: `sudo apt install realvnc-vnc-server`
- **Network performance tips**
  - If you experience lag or slow refresh rates, check your network quality.
  - Wired Ethernet generally offers the best performance.
- **Fixing display resolution issues**
  - If the VNC window appears too small or the resolution is incorrect, set a fixed resolution via: `sudo raspi-config` → **Display Options** → **VNC Resolution**
- **Ensure VNC is enabled**

  If VNC fails to connect, verify that it is enabled in: `sudo raspi-config` → `Interfacing Options` → `VNC`

- **Stopping the VNC service**

  To manually stop the VNC Server: `sudo systemctl stop vncserver-x11-serviced`

- **Security reminder**
  - VNC is designed for trusted local networks.
  - Do **not** expose VNC directly to the internet.
  - For secure remote access from outside your network, use **Raspberry Pi Connect** or a VPN.

CHAPTER

SIX

FAQ

6.1 Q1: What versions of PiDog are available?

PiDog comes in **Standard** and **V2** versions:

- **Standard Version**: Compatible with Raspberry Pi 3B+/4B/Zero 2W, **not** compatible with Raspberry Pi 5.
- **V2 Version**: Compatible with Raspberry Pi 3/4/5 and Zero 2W. It improves Robot HAT and servo driver circuits and provides better power support for Pi 5.
- **Power Supply**: V2 has enhanced power management for higher power consumption applications.

6.2 Q2: How do I install the required modules?

```
# Robot HAT
git clone -b 2.5.x https://github.com/sunfounder/robot-hat.git --depth 1
cd robot-hat && sudo python3 install.py

# Vilib
git clone https://github.com/sunfounder/vilib.git
cd vilib && sudo python3 install.py

# PiDog
git clone https://github.com/sunfounder/pidog.git --depth 1
cd pidog && sudo pip3 install . --break
```

If there's no sound:

```
# I2S Audio
cd ~/robot-hat
sudo bash i2samp.sh
```

Run multiple times if needed.

6.3 Q3: How do I run the first demo?

```
cd ~/pidog/examples
sudo python3 1_wake_up.py
```

PiDog will wake up, sit down, and wag its tail.

6.4 Q4: What built-in actions and sounds are available?

- Actions: `stand`, `sit`, `wag_tail`, `trot`, etc.
- Sounds: `bark`, `howling`, `pant`, etc.

Run:

```
sudo python3 2_function_demonstration.py
```

Enter numbers to trigger actions.

6.5 Q5: How does PiDog use sensors?

- **Ultrasonic**: Obstacle avoidance and patrol.
- **Touch**: Front touch = alert; back touch = enjoyment.
- **Sound Direction**: Responds to the direction of sound.

6.6 Q6: What AI features does PiDog support?

PiDog integrates with **TTS**, **STT**, and **LLM**:

- **TTS**: Espeak, Pico2Wave, Piper, OpenAI.
- **STT**: Vosk (offline).
- **LLM**: Ollama (local), OpenAI (online).

6.7 Q7: Do I need to calibrate the servos?

Yes — **servo calibration is required for both Standard and V2 versions** to ensure stable movement and prevent damage.

V2 Version

Press the **zeroing button** on the Robot HAT to automatically set all servos to 0°. This simplifies the zeroing process without running a script.

Standard Version

Run the zeroing script **before installation**:

```
cd ~/pidog/examples
sudo python3 servo_zeroing.py
```

After installation (both versions), manually verify and fine-tune servo angles to align each limb with the calibration ruler to avoid instability, blocking, or mechanical stress and ensure smooth walking and accurate posture control.

6.8 Q8: Why is my PiDog walking unstably?

- Confirm all servos were installed at 0°.
- Ensure servo angles match the calibration ruler (60°/90°).
- Check that the battery is fully charged.
- Tighten all servo screws.

6.9 Q9: Why is my camera not working?

- Ensure the camera cable is **firmly inserted** into the CSI interface and the black locking tab is secured.
- **Power off** the Raspberry Pi before plugging or unplugging the camera to prevent damage.
- Test the camera using `libcamera-hello` or `raspistill` to verify image output.
- Re-seat the cable if it is loose or improperly installed.

6.10 Q10: Why isn't the speaker working?

- Ensure the volume is not muted and the I2S audio driver is installed.
- If there's no sound, reconfigure I2S with the following:

```
cd ~/robot-hat
sudo bash i2samp.sh
```

- Restart the Raspberry Pi after running the script.

6.11 Q11: Why isn't the microphone working?

- Check whether the system recognizes the microphone with:

```
arecord -l
```

- Test the recording function with:

```
arecord -D plughw:1,0 -f cd test.wav
```

- If no audio is recorded, select the correct input device in the audio settings or use `alsamixer` to adjust input volume.
- Make sure no other process is occupying the audio input device.

6.12 Q12: Why isn't the sound direction sensor working?

- Ensure the sound direction sensor is connected to the correct SPI interface.
- Check that all cables are securely connected and not reversed.
- Make sure the power supply is stable and the sensor is not obstructed.
- Reboot the device and try running the sensor example script again.

6.13 Q13: Why doesn't the touch sensor respond?

- Ensure all touch sensor cables are firmly connected.
- Remember: a **LOW** signal means the sensor is being touched.
- Test the GPIO pin with `gpio readall` or Python code to confirm signal detection.
- Re-check wiring and orientation.

6.14 Q14: Why is the LED board not lighting up or blinking incorrectly?

- Verify the LED board is powered by **3.3V** and connected to the I2C port.
- Make sure **I2C is enabled** on the Raspberry Pi.
- Run the following command to check if the board is recognized:

```
i2cdetect -y 1
```

- If no device appears, recheck wiring and restart the Pi.

6.15 Q15: How does PiDog get power?

- Use a 5V 3A Type-C power adapter.
- Red light = charging, off = fully charged.
- You can power it while charging.
- If the indicator doesn't light, charge it first.

CHAPTER

SEVEN

COPYRIGHT NOTICE